PRO GRADU -TUTKIELMA

Rauno Paukkeri

Tilastolliset oppimisyhdistelmät asiakasvaihtuvuuden ennustamisessa

TAMPEREEN YLIOPISTO
Informaatiotieteiden yksikkö
Tilastotiede
Joulukuu 2013

Tampereen yliopisto
Informaatiotieteiden yksikkö
PAUKKERI, RAUNO: Tilastolliset oppimisyhdistelmät asiakasvaihtuvuuden ennustamisessa
Pro gradu -tutkielma, 42 s., 1 liites.
Tilastotiede
Joulukuu 2013

Tiivistelmä

Tämän tutkielman tarkoituksena on selvittää ja vertailla erilaisia tilastollisia oppimisyhdistelmiä erään suomalaisen teleoperaattorin asiakasvaihtuvuuden ennustamisessa. Tutkielmassa tarkastellaan myös, onko opetusaineiston painottamisella vaikutusta asiakasvaihtuvuuden ennustamisen tarkkuuteen. Asiakasvaihtuvuudella operaattorit tarkoittavat asiakkaan poistumista, joka ilmaistaan vuositasolla prosentteina. Prosenttiosuus kuvaa poistuvien asiakkaiden osuutta kaikista asiakkaista.

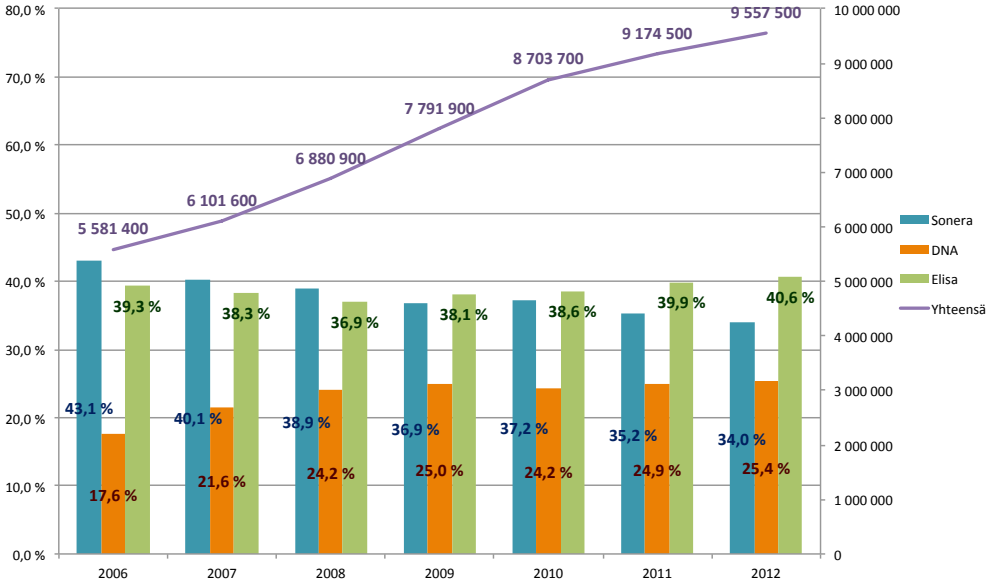
Asiakasvaihtuvuuden tuomien kustannusten vuoksi operaattorit pyrkivät kohdentamaan asiakassuhteiden hoitamista erityisesti vaihtumassa oleviin asiakkaisiin. Kohderyhmän löytäminen asiakaskannasta ei kuitenkaan ole helppoa, sillä asiakasvaihtuvuus on tilastollisesti hyvin haasteellinen tutkimuksen kohde. Asiakasvaihtuvuuden tarkastelu kuukausitasolla on haasteellista, koska vaihtuvuus on yleensä alle 2%. Tällöin yksittäisten tilastollisten mallien ongelmana on, että ne sovittuvat huonosti aineistoon. Ongelmaan on kehitetty erilaisia oppimisyhdistelmiä, joiden avulla saadaan parannettua yksittäisen mallin monimuotoisuutta ja ennusteiden tarkkuutta. Tutkielmassa vertailtiin erilaisia oppimisyhdistelmiä, erityisesti *Bagging* ja *Boosting*-algoritmeihin ja päätöspuihin perustuvia menetelmiä. Oppimisyhdistelmistä tutkielmaan valittiin *Bagging*-algoritmi yhdessä päätöspuiden kanssa, *Real AdaBoost*, *Gradient Boosting* sekä satunnaismetsä-algoritmit. Menetelmistä tehokkaimmiksi osoittautuivat *Gradient Boosting* ja satunnaismetsä riippuen käytettävän opetusaineiston painottamisesta.

Asiasanat *Bagging*, *Boosting*, *Real AdaBoost*, *Gradient Boosting*, satunnaismetsä, päätöspuut

Sisältö

1 Johdanto

Tietoliikenne ja ICT-palvelualat (*information and communications technology*) ovat 10 viime vuoden aikana kehittyneet huimaa vauhtia. Teknologia monipuolistuu ja mitä erilaisimpia palveluita tuodaan markkinoille. Tällä hetkellä suomalaisista kotitalouksista 99 prosentilla onkin vähintään yksi matkapuhelin ja älypuhelimiakin on 54 prosentilla kotitalouksista (Kuluttajabarometri 2013). Kuluttajat ovat siis ottaneet ICT-alan edut matkapuhelimien osalta erittäin laajalti käyttöön. Suomen Tilastokeskuksen mukaan Suomessa oli vuoden 2012 lopulla 5 426 674 asukasta. Matkapuhelinpalveluja tarjoavat operaattorit puolestaan mainitsevat vuosikertomuksissaan, että heillä on Suomessa 9 557 500 matkapuhelinliittymää vuonna 2012 (kuvio 1.1). Jokaista suomalaista kohden on siis noin 1,76 matkapuhelinliittymää. Näin ollen

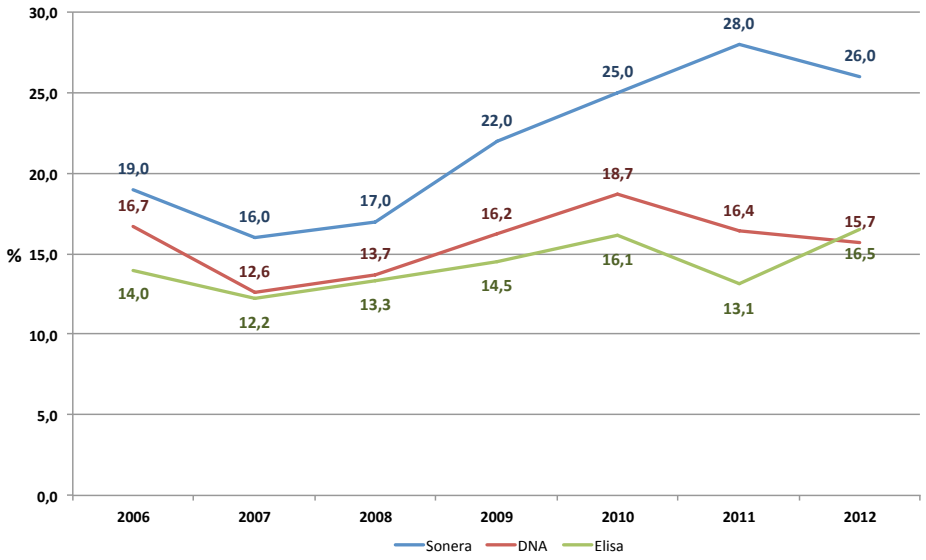


Kuvio 1.1. Matkapuhelinliittymien lukumäärä sekä kunkin operaattorin osuus liittymistä Suomessa vuosina 2006–2012 (Dna Oy 2006–2012; Elisa Oyj 2006–2012; TeliaSonera Finland Oyj 2006–2012).

kilpailu asiakkaista on kovaa, sillä liittymien lukumäärä asukasta kohden ei lisäänny rajattomasti. Tämä osaltaan lisää operaattoreille haasteita pitää yhä parempaa huolta asiakassuhteistaan, mikäli operaattori haluaa säilyttää tai jopa parantaa nykyistä markkinaosuuttaan. Kuten kaupankäynnissä yleensä, myös matkapuhelinpalveluja tarjoavilla operaattoreilla asiakaspoistuma on merkittävä kustannuskysymys, jossa jokaisen poistuneen asiakkaan tilalle hankittu uusi asiakas aiheuttaa operaattoreille huomattavan kuluerän. Asiakaspoistuman mallinnuksella pyritään vähentämään kustannuksia ja pitämään saavutettu markkinaosuus vähintäänkin samalla tasolla.

Mallintamisella yritys saa etukäteen tiedon mahdollisesti poistuvista asiakkaista ja sen perusteella voidaan tehdä tarvittavia toimenpiteitä asiakkuuden säilyttämiseksi.

Kun tarkastellaan operaattoreiden tämän hetkistä tilannetta Suomessa, kuviosta 1.1 nähdään, että vuoteen 2008 Sonera on hallinnut matkapuhelinliittymämarkkinoita. Tämän jälkeen Elisa on mennyt edelle DNA:n säilyttäessä reilun 24% markkinaosuuden. Kuviosta 1.2 puolestaan nähdään suomalaisten teleoperaattoreiden asiakasvaihtuvuus vuositasolla vuosina 2006–2012. Selvästikin Soneralla on ollut hieman korkeampi vaihtuvuus kuin kahdella muulla operaattorilla. Neljän edellisen vuoden aikana Soneran vaihtuvuus on noussut selvästi korkeammaksi kuin kilpailijoiden, mikä on osaltaan vaikuttanut operaattoreiden valtasuhteisiin Suomessa (vrt. kuvio 1.1).



Kuvio 1.2. Suomessa toimivien operaattoreiden asiakaspoistumaprosentit vuosina 2006–2012 (Dna Oy 2006–2012; Elisa Oyj 2006–2012; TeliaSonera Finland Oyj 2006–2012).

Tämän tutkielman tavoitteena on tarkastella operaattorin asiakkaan poistuman ennustamista etukäteen liittymätietojen ja asiakkaan käyttäytymisen perusteella. Tämän päivän markkinastrategiassa vanhan asiakkaan pitäisi antaa yritykselle jotakin todellista lisäarvoa. Tämän vuoksi työssä pyritään myös tunnistamaan ja ottamaan huomioon asiakkaan arvokkuus eli se, miten paljon asiakas tuottaa yritykselle voittoa. Tutkielmassa vertaillaan erilaisia tilastollisia luokittelualgoritmeja ja sovelletaan niitä asiakaspoistuman mallintamiseen. Tarkoituksena tässä työssä on tutkia jo olemassa olevien algoritmien toimivuutta suuressa aineistomassassa ja löytää paras algoritmi, jolla jatkossa voitaisiin parhaiten mallintaa kunkin asiakkaan poistumatodennäköisyyttä.

Algoritmien tarkoituksena on löytää luokittelijan avulla aineistolle esitystapa,

jonka avulla poistuvat asiakkaat sekä pysyvät asiakkaat voidaan erottaa toisistaan. Luokittelun onnistuminen riippuu siitä, miten suuri poistuvien sekä pysyvien asiakkaiden sisäinen vaihtelu on verrattuna niiden väliseen vaihteluun. Aineiston käyttöä luokittelijan muodostamiseksi kutsutaan opettamiseksi. Luokittelija käyttää oppimiseen opetusaineistoa, joka muodostuu esimerkkihavainnoista. Esimerkkihavainnot poimitaan puolestaan alkuperäisestä aineistosta erilaisilla menetelmillä riippuen luokittelualgoritmista. Luokkien taustalla olevat tuntemattomat parametrit estimoidaan opetusaineiston avulla. Luokittelijan arviointi suoritetaan luokitteluvirheen perusteella, joka saadaan laskemalla esimerkiksi testiaineiston havaintojen väärinluokiteltujen yksilöiden osuus.

Tutkielma etenee siten, että luvussa 2 esitellään ja kuvaillaan käytössä olevaa aineistoa. Tämän jälkeen luvussa 3 kerrotaan, mitä on tilastollinen oppiminen. Luvussa käsitellään myös tutkielmassa käytettäviä luokittelualgoritmeja sekä menetelmiä, joiden perusteella vertaillaan, mikä algoritmeista soveltuu parhaiten tämän aineiston kaltaiseen tilanteeseen. Algoritmien tuottamia tuloksia ja menetelmien vertailua käsitellään puolestaan luvussa 4. Lopuksi luvussa 5 kootaan tutkielman tulokset yhteen.

2 Aineiston kuvaus

Tässä luvussa esitellään tutkielmassa käytettävä tutkimusaineisto. Lisäksi käsitellään sitä, miten puuttuvia havaintoja on käsitelty erilaisten muuttujien kohdalla. Jotta muuttujien sisältämä informaatio ei olennaisesti muutu, käsitellään numeeristen ja luokitteluasteikollisten muuttujien puuttuvat arvot eri tavoin.

2.1 Aineisto

Tutkielmassa aineistoina käytetään kahta erään suomalaisen operaattorin tarjoamaa otosta asiakastietokannasta. Aineistot ovat täysin anonymisoituja, ja niitä on käytetty vain tilastollista analysointia varten. Ensimmäinen aineisto sisältää operaattorin 79 864 liittymää, joista poistuneita on 1 216 eli noin 1,5% kaikista liittymistä. Toinen aineisto on puolestaan tasapainotettu aineisto, missä on yhtä monta poistunutta kuin aktiivistakin asiakasta. Jälkimmäinen aineisto sisältää 79 210 liittymää, jolloin poistuneita ja aktiivisia asiakkaita on 39 605 kappaletta.

Molemmissa aineistoissa on 70 asiakasta kuvailevaa muuttujaa, joista luokitteluasteikollisia muuttujia on 44 ja numeerisia muuttujia 26. Luokitteluasteikolliset muuttujat sisältävät tietoja esimerkiksi asiakkaan poistumisesta, päätelaitteesta, asuinpaikkakunnasta sekä erilaisista puhelimen käyttöön liittyvistä luokittelijoista. Numeeriset muuttujat puolestaan kertovat esimerkiksi asiakkaan, päätelaitteen ja sopimuksen iän. Lisäksi aineistossa on numeerisia muuttujia, jotka kuvaavat puhelimen keskimääräisiä käytön määriä 3, 6 ja 12 viime kuukauden aikana. Nämä muuttujat sisältävät tietoja puheluminuuteista, puheluiden ja tekstiviestien lukumääristä sekä datan käytön määrästä.

Aineistoissa kustakin puhelimen käyttöön liittyvästä muuttujasta on tehty aina kaksi erilaista muuttujaa siten, että toinen niistä on numeerinen ja toinen luokitteluasteikollinen muuttuja. Luokitteluasteikollinen muuttuja jakaa jatkuvan muuttujan sisältämän tiedon viiteen eri luokkaan puhelimen käytön perusteella. Aineistossa on myös muuttujia, jotka kertovat, miten asiakkaan puhelimen käyttö on viimeisimmän kuun aikana muuttunut verrattuna aikaisempiin kuukausiin. Tämä on toteutettu siten, että tarkastellaan asiakkaan puhelimen käyttöä neljän viime kuukauden aikana. Näistä kuukausista kolmen vanhimman kuukauden keskiarvo vähennetään viimeisen kuukauden keskiarvosta. Näin saadaan muuttujat, joilla voidaan tarkastella puhelimen käytön muutoksia. Jos muuttujan saama arvo on lähellä nollaa tai arvo on positiivinen, puhelimen käytössä ei ole muutoksia tai käytön määrä on kasvanut. Jos taas arvo on negatiivinen, puhelimen käyttö on vähentynyt. Tarkemmat kuvaukset tutkielmassa käytetyistä muuttujista löytyvät liitteestä.

2.2 Puuttuvat havainnot

Alkuperäinen aineisto on hyvin laaja. Siitä joudutaan kuitenkin rajaamaan muuttujia pois puuttuvien havaintojen runsauden vuoksi. Karsiminen on tehty, jotta puuttuvat havainnot eivät vääristäisi analyysin tuloksia. Aineiston muuttujat, joissa on yli 30% puuttuvia havaintoja, on jätetty pois lopullisesta mallinnuksesta. Näin muuttujia karsimalla mallinnukseen jää 56 muuttujaa, joista 30 muuttujaa on luokitteluasteikollisia ja 26 jatkuvia.

Jäljelle jäävien muuttujien puuttuvia havaintoja on käsitelty kahdella eri tavalla. Luokitteluasteikollisten muuttujien puuttuville havainnoille on tehty oma luokka, joka mallinnuksessa kertoo, että havainto on ollut puutteellinen. Jatkuvien muuttujien puuttuviin havaintoihin on puolestaan asetettu kunkin muuttujan keskiarvo, joka on laskettu kyseisen muuttujan havainnoista siten, että puuttuvat havainnot on jätetty ottamatta huomioon.

3 Analyysimenetelmät

Luvussa käsitellään tutkielmassa tarkasteltuja tilastollisia oppimismenetelmiä, erityisesti luokitteluun soveltuvia oppimisyhdistelmiä. Tutkielmassa tarkoituksena on vertailla erilaisia oppimisyhdistelmiä keskenään ja valita niistä paras menetelmä asiakaspoistuman mallinnukseen. Käytetyiksi menetelmiksi valittiin *Boosting*-algoritmiin perustuvat *Real AdaBoost* ja *Gradient Boosting* -algoritmit, satunnaismetsä (*Random Forest*) sekä *Bagging*-algoritmi yhdessä päätöspuiden kanssa. Menetelmien lisäksi luvussa perehdytään myös muuttujien valintaan sekä menetelmien tehokkuuden arviointimenetelmiin. Luvussa käsitellyistä analyysimenetelmistä saa lisätietoa esimerkiksi teoksesta *The Elements of Statistical Learning: Data Mining, Inference and Prediction* (Hastie, Tibshirani & Friedman, 2011).

3.1 Tilastollinen oppiminen

Tilastollinen oppiminen (*Statistical learning*) on mentelmä, jossa analysoidaan aineistoja ja pyritään oppimaan analyysin pohjalta kuvauksia eri datapisteiden välillä. Menetelmän tärkeimpänä ominaisuutena on se, että malli kehittyy analyysin edetessä. Tilastollinen oppiminen voidaan jakaa karkeasti luokitellen kahteen eri osa-alueeseen: ohjattuun oppimiseen sekä ohjaamattomaan oppimiseen. Tyypillisin ero näiden osa-alueiden välillä on se, että ohjatuissa menetelmissä käytetään nimettyä opetusaineistoa mallin rakentamiseen, kun taas ohjaamattomissa menetelmissä ei käytetä nimeämistä. Tämän lisäksi on olemassa variaatioita, jotka sisältävät piirteitä niin ohjatuista kuin ohjaamattomista menetelmistä. Näistä hyviä esimerkkejä ovat puoliksi ohjattu oppiminen, heikosti ohjattu oppiminen sekä vahvistusoppiminen.

3.1.1 Ohjattu oppiminen

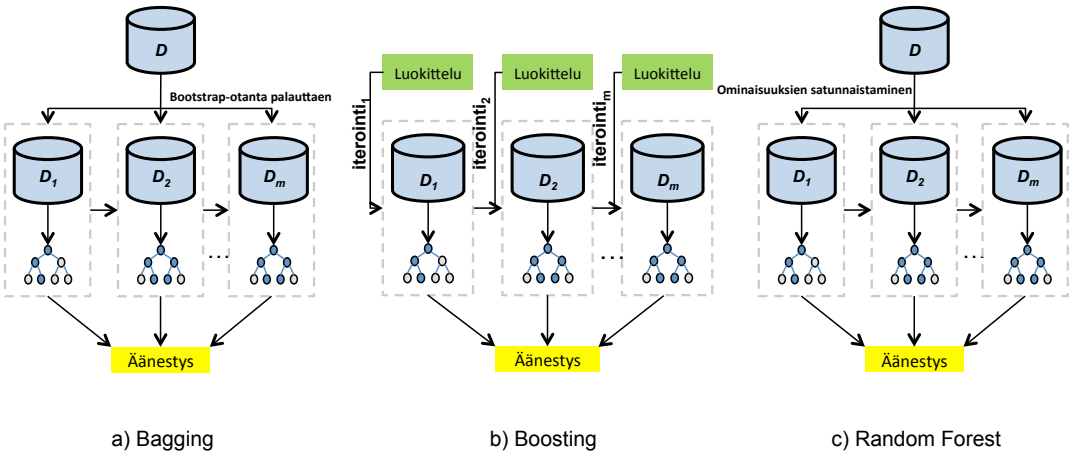
Ohjatussa oppimisessa (*Supervised learning*) mallin parametrit sovitetaan nimettyihin datakokonaisuuksiin. Nimet lisätään yleensä manuaalisesti aineistoon ja ne näyttävät, millainen struktuuri aineistoista pitäisi löytyä. Ohjattu oppiminen jakautuu kahteen eri tyyppiin: luokitteluun, jossa aineisto luokitellaan erillisiin ennalta nimettyihin luokkiin, sekä regressioanalyysiin, jota käytetään selitettävän muuttujan ollessa jatkuva. Ohjatun oppimisen menetelmät opetetaan ensin opetusaineistolla, jonka jälkeen menetelmiä voidaan käyttää uuden nimeämättömän aineiston nimeämiseen. Yksi tunnetuimmista ohjatun oppimisen menetelmistä on tukivektorikone (*support vector machine*), joka on luokitteluun ja käyränsovitustehtäviin sopiva lineaarinen luokitinmalli. Lisäksi on olemassa useita yhdistelmäoppimisen kokonaisuuteen kuuluvia algoritmeja, joista kuviossa 3.1 on graafisesti kuvattuna kolme eri algoritmia: *Boosting*-meta-algoritmi, bootstrap aggregointia soveltava erottelufunktio (*Bagging*) sekä siihen perustuva uudempi satunnaismetsä-algoritmi. Näitä kolmea yhdistelmäalgoritmia käsitellään tarkemmin tässä tutkielmassa. (Hastie et al. 2011)

Ohjatun oppimisen menetelmät ovat erittäin hyviä erilaisissa luokitteluongelmissa, joissa opetusaineiston kohdeluokittelu tunnetaan ja kun opetusaineisto on riittävän laaja, laadukas ja luokiteltava. Tällöin menetelmät voivat oppia melko oikean kuvauksen luokkien ja datapisteiden välillä.

3.1.2 Ohjaamaton oppiminen

Ohjaamattoman oppimisen (*Unsupervised learning*) menetelmien tarkoituksena on löytää nimeämättömän aineiston piilotettu rakenne. Tällaisia menetelmiä ovat esimerkiksi k keskiarvon menetelmä, itseorganisoituva kartta sekä riippumattomien komponenttien analyysi.

Ohjaamattoman oppimisen erityisenä etuna on, että nimeämättömiä tai uudelleen nimettyjä aineistoja voidaan hyödyntää analyysissä. Jos tiedetään luokitteluongelman kohde, ohjatun oppimisen menetelmillä saavutetaan parempi virheettömyys kuin ohjaamattoman oppimisen menetelmillä. Kuitenkin joissain tilanteissa oppimattomat menetelmät toimivat yhtä hyvin tai jopa paremmin kuin ohjatut menetelmät, koska niillä on laajat sovellusmahdollisuudet ja kyky yleistää aineistoa. Täysin automaattiset algoritmit käyttävät pääsääntöisesti ohjaamatonta oppimista, koska menetelmien sovelluksissa yleensä suljetaan pois mahdollisuus ohjattujen menetelmien käyttöön. Tämä johtuu siitä, että automaattiset algoritmit käyttävät pääsääntöisesti nimeämättömiä aineistoja. (Paukkeri, Väyrynen & Arppe, 2012)



Kuvio 3.1. *Bagging*, *Boosting* ja satunnaismetsä (*Random Forest*) -algoritmit kaaviolla kuvattuna.

3.2 Yhdistelmäoppiminen

Ohjatun oppimisen kokonaisuuteen kuuluvan yhdistelmäoppimisen (*ensemble learning*) tarkoituksena on parantaa yksittäisen tilastollisen menetelmän sovittumista käytettävään aineistoon. Tutkielman kaltaisen luokitteluongelman tilanteessa oppimisyhdistelmissä käytetään yleensä yhtä perusluokittelijaa, jota sovitetaan usei-

ta kertoja aineistoon. Luokittelijan sovituksista saadaan yksittäiset hypoteesit, jotka yhdistetään yhdeksi hypoteesiksi. Saadun hypoteesin perusteella voidaan tehdä yksittäisiä ennusteita, jotka ovat usein luotettavampia kuin yksittäisiin tilastollisiin menetelmiin perustuvat ennusteet. Yhdistelmäoppimisen algoritmeja voidaan käyttää luokitteluongelmien lisäksi esimerkiksi käytetyn mallin päättelyn luotettavuuden arvioinnissa, muuttujien valinnassa, optimaalisuuden (tai lähes optimaalisuuden) parantamisessa, aineistojen fuusioimisessa, inkrementaalisessa ja ei-stationaarisessa oppimisessa sekä virheen arvioinnissa. (Opitz & Maclin, 1999)

Oppimisyhdistelmiä käytettäessä tarvitaan yleensä yksittäisiä malleja enemmän laskentaa. Tämän vuoksi perusluokittelijoina käytetään usein nopeita algoritmeja kuten esimerkiksi päätöspuita. Tutkielmassa käytettävät oppimisyhdistelmät perustuvat *CART* (*Classification and Regression Trees*) -algoritmeihin eli luokittelu- ja regressiopuihin, joista luokittelupuita käsitellään myöhemmin tarkemmin. Tutkielmassa käytettyjen *Bagging* ja *Boosting*-algoritmien lisäksi suosituimpia oppimisyhdistelmiä ovat esimerkiksi Bayesin optimaalinen luokittelija (*Bayes optimal classifier*) ja bayesiläinen mallin keskiarvostaminen (*Bayesian model averaging*).

3.2.1 Opetus- ja testiaineisto

Oppimisyhdistelmien algoritmeissa käytettävä aineisto jaetaan yleensä opetus- ja testiaineistoon jo ennen kuin käytettävää yhdistelmää sovelletaan aineistoon. Näin saadaan tarvittava aineisto oppimisyhdistelmän opettamiseen sekä luotettava testiaineisto esimerkiksi eri oppimisyhdistelmien vertailuun.

Oppimisyhdistelmät tarvitsevat usein opetusaineiston lisäksi aineiston, jonka avulla yhdistelmässä käytettävien menetelmien tehokkuutta voidaan arvioida. Menetelmien arviointia ei kannata tehdä oppimisyhdistelmien vertailuun käytettävällä testiaineistolla, koska tällöin on mahdollista aliestimoida oppimisyhdistelmän ennustevirhettä. Jos käytettävissä on rikas data-aineisto, paras lähestymistapa ongelmaan on jakaa aineisto satunnaisesti kolmeen luokkaan: opetus-, validointi- ja testiaineistoon. Tällöin oppimisyhdistelmässä opetusaineistolla muodostetaan mallin luokittelusääntö ja validointiaineistoa käytetään mallinvalintaan estimoimalla mallin ennustevirhettä. Testiaineistolla puolestaan arvioidaan oppimisyhdistelmän ennustevirhettä. On vaikea määritellä tarkasti, miten aineisto kannattaa jakaa kolmeen osaan, koska se riippuu aineiston signaalikohinasuhteesta ja opetusaineiston koosta. Tyypillisesti aineisto jaetaan kuitenkin siten, että opetusaineisto on 50% ja validointi- sekä testiaineistot ovat molemmat 25% alkuperäisestä aineistosta. (Hastie et al. 2011)

Aineiston jakaminen kolmeen osaan ei aina ole mahdollista. Siksi on kehitetty useita erilaisia menetelmiä, joilla voidaan arvioida mallin ennustevirhettä. Yksi toimiva menetelmä on seuraavassa luvussa esiteltävä bootstrap-menetelmä, jossa ennustevirhettä arvioidaan bootstrap-otosten avulla. Tutkielmassa käytettävät yhdistelmäoppimisalgoritmit soveltavat juuri bootstrap-menetelmää.

3.2.2 Bootstrap

Bootstrap-menetelmä on tilastollinen otantamenetelmä, jonka ajatuksena on korvata teoreettiset päätelmät useasti toistetuilla empiirisillä päätelmillä. Menetelmän avulla voidaan esimerkiksi tutkia käytetyn estimaatin virhettä.

Efronin (1979) kehittämää bootstrap-prosessia käytetään yleisimmin otantaan perustuvissa tilastollisissa menetelmissä, joissa poimitaan palauttaen otoksia alkuperäisestä aineistosta. Prosessin tarkoituksena on arvioida opetusaineiston tilastollista tarkkuutta estimaatilla $S(Z)$. Oletetaan, että tutkittava aineisto on muotoa $Z_i = (\boldsymbol{x}_i, y_i)$, missä y_i on ennustettava muuttuja ja $i = 1, 2, \ldots, N$. Bootstrap-menetelmässä poimitaan palauttaen otos $Z_l^{*b} = (Z_1^{*b}, Z_2^{*b}, \ldots, Z_L^{*b})$ aineistosta Z_i, missä $L \leq N$. Poiminta toistetaan B kertaa, jolloin saadaan poimittua otokset $Z^{*b} = (Z^{*1}, Z^{*2}, \ldots, Z^{*B})$. Jokaisesta otoksesta Z^{*b} lasketaan haluttu estimaatti $S(Z^{*b})$. Tämän jälkeen saaduista estimaateista $S(Z^{*b}),\ b = 1, 2, \ldots, B$, lasketaan lopullinen bootstrap-estimaatti $S(Z)$.

Esimerkiksi tutkielmassa käytetyssä *Bagging*-algoritmissa otanta tehdään bootstrap-menetelmän avulla. Koska menetelmässä otanta suoritetaan palauttaen, jokaisella aineiston havainnolla on yhtä suuri todennäköisyys tulla valituksi otokseen. Kun poimitaan L kappaleen bootstrap-otos aineistosta, jokaisen havainnon i keskimääräinen todennäköisyys tulla valituksi otokseen on

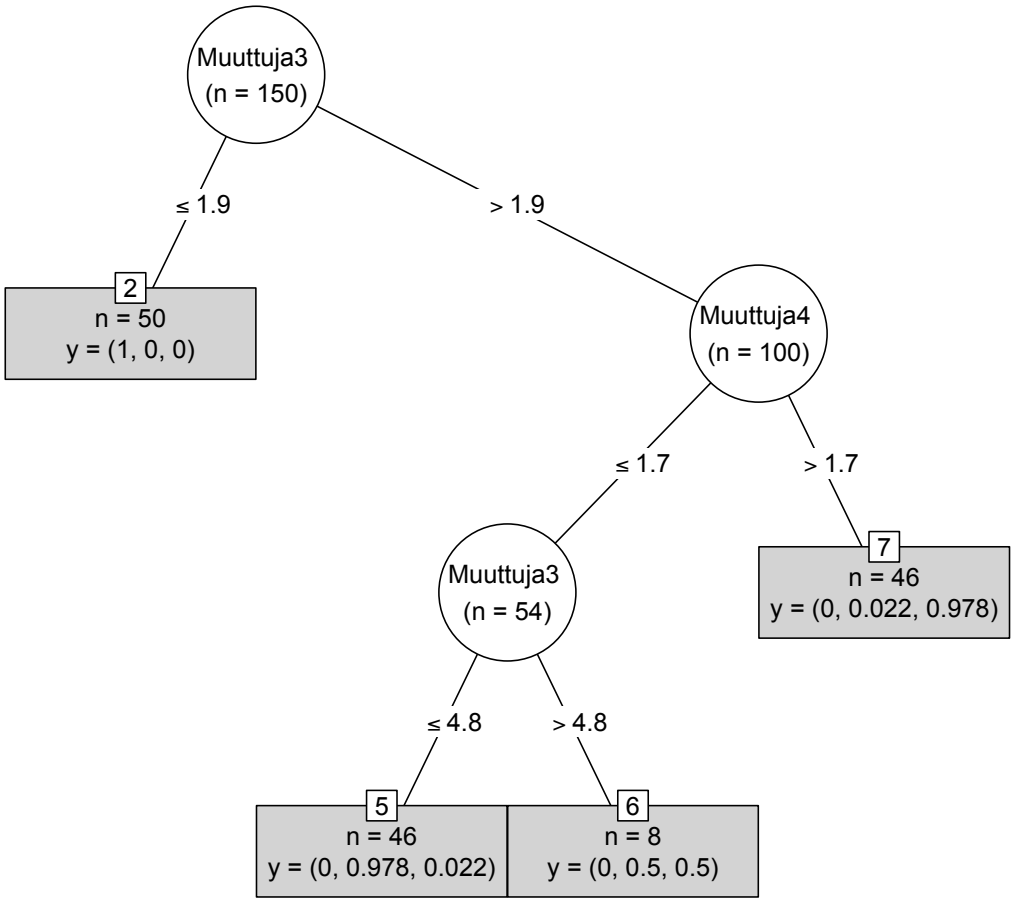
$$1 - \left(1 - \frac{1}{L}\right)^L \approx 1 - e^{-1} = 0,632. \tag{3.1}$$

Tämä mahdollistaa sen, että bootstrap-otannan avulla voidaan parantaa luokittelun tarkkuutta epävakaissa luokittelumenetelmissä kuten päätöspuissa tai neuroverkoissa. Näissä menetelmissä opetusaineiston ja puun rakenteen pienetkin muutokset voivat aiheuttaa merkittäviä eroja luokittelun tulokseen. (Breiman 1996)

3.2.3 Päätöspuut

Puihin perustuvat menetelmät ovat käsitteellisesti yksinkertaisia ja tehokkaita menetelmiä, joilla pystytään mallintamaan monimutkaisia epälineaarisia päätösrajoja. Näistä suosituimpia menetelmiä ovat luokittelu- ja regressiopuut (*CART*), joista seuraavaksi käsitellään luokittelupuita.

Luokittelu- eli päätöspuu on luokittelualgoritmi, jolla aineistoa voidaan pilkkoa sen piirteiden perusteella osajoukkoihin. Puut muodostuvat solmuista (*node*), oksista (*branch*) sekä juuresta (*root*). Puun juuri on algoritmin ensimmäinen luokittelumuuttuja, joka voidaan itse määrittää tai antaa algoritmin satunnaisesti valita. Yleensä juureksi valitaan muuttuja, joka jakaa opetusaineiston mahdollisimman täydellisesti tutkittavan luokkamuuttujan mukaisiin luokkiin. Aineisto jaetaan juurisolmussa olevan muuttujan arvojen mukaisesti eri luokkiin, jotka muodostavat puun ensimmäiset oksat (ks. kuvio 3.2). Jokaisesta oksasta alkaa puolestaan uusi osapuu, jolle toistetaan sama menettely. Aineiston jakamista oksiin jatketaan, kunnes uloimmilla oksilla on vain samaan luokkaan kuuluvia havaintoja. Näitä päätesolmuja kutsutaan puun lehdiksi, jotka kertovat päättelyn tuloksen. Päätöspuulla ennustettaessa uudet



Kuvio 3.2. Esimerkki eräästä luokittelupuusta, jossa ympyrät kuvaavat puun solmuja ja laatikot puun lehtiä. Solmujen ja lehtien välissä olevat viivat kuvaavat puun oksia.

havainnot jaetaan päätesolmuihin puussa olevan jaottelun mukaisesti, jolloin tiedetään, mihin luokkaan uudet havainnot kuuluvat.

Käytettäessä päätöspuihin perustuvia oppimisyhdistelmiä, on tärkeää ymmärtää, miten päätöspuita voidaan verrata keskenään. Merkitään, että y_i on aineiston ennustettava havainto ja vektori $\boldsymbol{x}_i$ on sitä vastaavien selittävien muuttujien vektori, missä $i = 1, 2, \ldots, N$. Merkitään myös, että k on ennustettavan muuttujan luokkamuuttuja, joka saa arvot $1, 2, \ldots, K$. Tällöin puun solmukohdassa m luokan k havaintojen suhteellinen osuus on

$$\hat{p}_{mk} = \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i = k), \tag{3.2}$$

missä I on indikaattorifunktio, joka saa arvon 1, kun havainto y_i kuuluu luokkaan

k ja muuten se saa arvon 0. Merkitään, että R_m on mahdollisten havaintojen joukko ja N_m on havaintojen lukumäärä solmukohdassa m. Kussakin solmukohdassa m aineiston havainnot luokitellaan enemmistöluokkaan kaavalla

$$k(m) = \arg\max_k \hat{p}_{mk}. \tag{3.3}$$

Tällöin kunkin solmukohdan m luokittelulle voidaan laskea luokitteluvirhe ja Gini-indeksi:

$$\begin{array}{ll} \text{Luokitteluvirhe:} & \frac{1}{N_m} \sum_{x_i \in R_m} I(y_i \neq k(m)) = 1 - \hat{p}_{mk}(m) \\ \text{Gini-indeksi:} & \sum_{k \neq k'} \hat{p}_{mk} \hat{p}_{mk'} = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}). \end{array} \tag{3.4}$$

Oletetaan, että poistuma-muuttujan kaltaisessa kahden luokan tilanteessa p on toisen luokan osuus aineistossa, jolloin luokitteluvirhe ja Gini-indeksi kaavasta (3.4) voidaan kirjoittaa yksinkertaisemmassa muodossa:

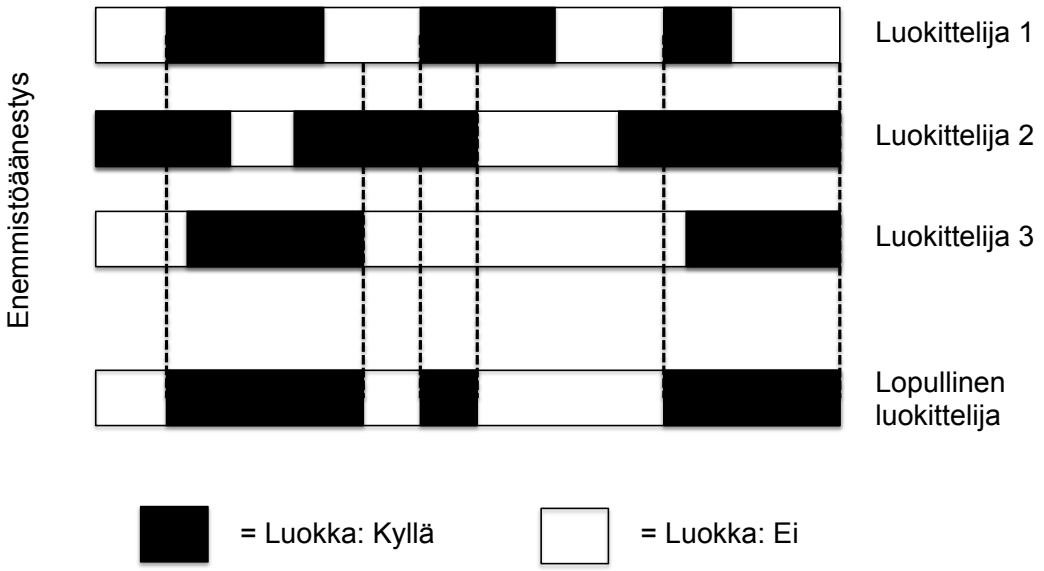
$$\begin{array}{ll} \text{Luokitteluvirhe:} & 1 - \max(\hat{p}, 1 - \hat{p}) \\ \text{Gini-indeksi:} & 2\hat{p}(1 - \hat{p}). \end{array} \tag{3.5}$$

Menetelmistä Gini-indeksi on herkempi solmun todennäköisyyksien muutokselle kuin luokitteluvirhe. Tämän vuoksi Gini-indeksi on usein luokitteluvirhettä parempi menetelmä luokittelupuun kasvatuksessa, sillä herkemmällä todennäköisyyksien muutosten analysoinnilla saavutetaan yleensä parempia päättelytuloksia. Lisäksi se on derivoituva, ja siten sopivampi numeeriseen optimointiin. (Hastie et al. 2011, luku 9.2.3)

3.3 *Bagging*

Bagging (***b**ootstrap **agg**regat**ing***) on bootstrap-aggregointiin perustuva luokittelualgoritmi, joka on yksi varhaisimmista ja mahdollisesti yksinkertaisimmista yhdistelmäoppimisen algoritmeista. Menetelmässä jokaisella iterointikierroksella perusluokittelijaa opetetaan opetusaineistosta bootstrap-otannalla poimituilla otoksilla. Tämän jälkeen opetetut luokittelijat yhdistetään yhdeksi luokittelijaksi enemmistöäänestyksellä, jossa lopullinen luokittelija muodostetaan kaikkien luokittelijoiden luokkajaon enemmistön perusteella (ks. kuvio 3.3). Enemmistöäänestyksen perusteella saadulla luokittelijalla voidaan tehdä opetusaineistoon perustuvia ennusteita. *Bagging*-luokittelun monimuotoisuus saavutetaan siis poimimalla palauttaen bootstrap-otoksia opetusaineistosta. Yksinkertaisuudesta huolimatta Breimanin (1996) kehittämä algoritmi toimii erityisen hyvin, kun käytetään menetelmiä, joissa on korkea varianssi ja pieni harha. Esimerkkinä näistä menetelmistä mainittakoon päätöspuut, joita käytetään tutkielman *Bagging*-algoritmissa (Breiman, Friedman, Olshen & Stone, 1984). Lisäksi tutkielmassa käytettävä satunnaismetsä-algoritmi perustuu osittain bootstrap-aggregointiin.

Seuraava teoriaosuus mukailee Breimanin (1996) esitystä *Bagging*-algoritmista. Merkitään, että opetusaineisto on muotoa $L = (L_1, L_2, \ldots, L_N)$ ja opetusaineiston



Kuvio 3.3. Kuviossa yksinkertainen esimerkki kolmen luokittelijan perusteella tehdystä enemmistöäänestyksestä binäärissä luokittelutilanteessa. Äänestyksen perusteella saadaan lopullisen luokittelijan muodostama luokkajako.

yksittäinen havainto on $L_i = (\boldsymbol{x}_i, y_i)$, missä $y_i \in \{0, 1\}$ tarkoittaa selitettävää muuttujaa eli asiakkaan poistumaa ja N opetusaineiston havaintojen lukumäärää. Vektori $\boldsymbol{x}_i = (x_{i1}, x_{i2}, \ldots, x_{ip})$ on puolestaan p-dimensioinen selittävä muuttujavektori, joka sisältää p selittävän muuttujan arvot havainnolla i. Jokaisella iterointikierroksella $b = 1, 2, \ldots, B$ opetusaineistosta L poimitaan satunnaisesti palauttaen N kappaleen bootstrap-otos L^{*b}. Kukin otos siis sisältää saman määrän havaintoja kuin opetusaineisto, mutta kaavan (3.1) mukaisesti jotkin havainnot voivat tulla valituksi otokseen useammin kuin kerran ja vastaavasti osa havainnoista jää kokonaan otosten ulkopuolelle. Jokaisella iterointikierroksella bootstrap-otokselle L^{*b} estimoidaan päätöspuulla $\hat{f}$ bootstrap-ennuste, jolloin iterointien päätteeksi on yhteensä B kappaletta pistefunktioita $\hat{f}^{*1}(x), \hat{f}^{*2}(x), \ldots, \hat{f}^{*B}(x)$. Nämä funktiot aggregoidaan lopulliseksi pistemääräksi kaavalla

$$\hat{f}_{bag}(x) = \frac{1}{B} \sum_{b=1}^{B} \hat{f}^{*b}(x). \tag{3.6}$$

Pisteiden avulla voidaan tehdä lopullinen luokittelu, kun käytetään kaavaa

$$\hat{c}_{bag}(x) = \text{sign}(\hat{f}_{bag}(x) - \tau_B), \tag{3.7}$$

jossa funktio $\text{sign}(\hat{f}_{bag}(x) - \tau_B)$ palauttaa arvon -1, kun $\hat{f}_{bag}(x)$ on raja-arvoa τ_B pienempi tai arvon 1 päinvastaisessa tilanteessa.

Tilanteessa, jossa asiakkaan poistuma-arvo on tuntematon, yleinen bootstrap-otosten pohjalta opetettu luokittelija $\hat{f}$ palauttaa jokaiselle asiakkaalle tunnusomainen arvon $\hat{f}(x)$. Tätä arvoa voidaan pitää kuhunkin asiakkaaseen liittyvänä pistemääränä, joka kuvastaa asiakaspoistumaan liittyvää riskiä. Kahden ryhmän tilanteessa

luokittelija lasketaan kaavalla

$$\hat{c}(x) = \text{sign}(\hat{f}(x) - \tau_B). \tag{3.8}$$

Jos $\hat{f}(x_i)$ on suurempi kuin τ_B, niin asiakas i luokitellaan poistuvaksi. Kun taas $\hat{f}(x_i)$ on pienempi kuin τ_B, niin ennustetaan, että asiakas i pysyy asiakkaana. Käytettäessä päätöspuuta, pisteet annetaan kaavalla

$$\hat{f}(x) = 2\hat{p}(x) - 1, \tag{3.9}$$

missä $\hat{p}(x)$ on päättelypuilla estimoitu poistumatodennäköisyys. Luonnollisin arvo τ_B:lle on 0, joka on funktion $\hat{f}(x)$ saamien arvojen -1 ja 1 puolessa välissä.

Bagging-algoritmi siis yksinkertaistettuna tuottaa B kappaletta uudelleenpoimittuja opetusaineistoja, joihin on sovellettu luokittelumenetelmää. Nyt voidaankin pohtia, mikä on hyvä valinta vakiolle B eli kuinka monta bootstrap-otosta poimitaan, jotta saadaan mahdollisimman vähäisellä otosten määrällä paras mahdollinen *Bagging*-ennuste. Strategisesti järkevintä on käyttää opetusaineiston määrittelykriteerinä luokitteluvirhettä. Tällöin B valitaan siten, että ennustevirheen arvo iteraatioiden B ja $B + 1$ välillä on merkityksetön. Virheiden minimoimiseen riittää yleensä noin 50 bootstrap-kierrosta (B). Kierrosten lukumäärä riippuu kuitenkin otoksen koosta ja laskentakustannuksista ennusteen arvioinnissa (Breiman 1996).

3.4 *Boosting*

Boosting on yksi tehokkaimmista käyttöön otetuista tilastollisista oppimisalgoritmeista viimeisen kahdenkymmenen vuoden aikana. Kuten *Bagging*, myös *Boosting* on uudelleenotantaan perustuva luokitteluyhdistelmä, jossa otannat yhdistetään enemmistöäänestyksellä (Hastie et al. 2011, s. 337). *Boosting*-algoritmissa otosten poiminta on suunnattu antamaan perättäisille luokittelijoille kaikkein oleellisimmat tiedot opetusaineistosta.

Oletetaan, että käytettävä aineisto on muotoa $Z = \{Z_1, Z_2, \ldots, Z_N\}$ ja $Z_i = (\boldsymbol{x_i}, y_i)$. Yleisesti voidaan ajatella, että jokaisen *Boosting*-meta-algoritmin iterointikierros koostuu kolmesta eri luokittelijasta C_i. Aluksi poimitaan palauttamatta L_1 ($< N$) havainnon otos Z_1^* aineistosta Z. Ensimmäinen luokittelija, C_1, opetetaan saadulla otoksella Z_1^*. Poimitaan uudelleen L_2 ($< N$) havainnon otos Z_2^* aineistosta Z, siten, että otokseen valitaan ne havainnot, jotka luokittelija C_1 on luokitellut väärin. Tämän jälkeen opetetaan luokittelija C_2 otoksen Z_2^* havainnoilla. Valitaan vielä otokseen Z_3^* ne havainnot, joissa luokittelijat C_1 ja C_2 ovat eri mieltä ja opetetaan luokittelija C_3 näillä havainnoilla. Lopuksi saadut kolme luokittelijaa yhdistetään enemmistöäänestyksellä.

3.4.1 *Real AdaBoost*

AdaBoost (**Ada**ptive **boost**ing) on Freundin ja Schapiren (1995) kehittämä tilastolliseen yhdistelmäoppimiseen kuuluva algoritmi, jossa *Boosting*-menetelmän avulla ratkotaan luokittelu- ja regressio-ongelmia. *AdaBoost*-algoritmista on olemassa

monta eri variaatiota. Mainittakoon esimerkkeinä *AdaBoost.M2* (luokittelu), *AdaBoost.R* (regressio), moniluokkaisiin luokitteluongelmiin tarkoitettu *Adaboost.M1* sekä tutkielmassa käytetty binäärisiin luokitteluongelmiin tarkoitettu *Real AdaBoost* -algoritmit.

Yleisesti *Boosting*-menetelmissä sovelletaan järjestelmällisesti heikkoa oppijaa adaptiivisesti painottuviin versioihin alkuperäisestä otoksesta. *AdaBoost.M1*-menetelmässä on mahdollista tuottaa ainoastaan binäärisiä luokittelusääntöjä. Toisin kuin *AdaBoost.M1*-menetelmässä, vuonna 1999 Schapiren ja Singerin kehittämä *Real AdaBoost* -algoritmi antaa mahdollisuuden tehdä myös yksittäisille havainnoille pisteennusteita, mikä mahdollistaa yksittäisten havaintojen tarkastelun mallinnuksen pohjalta.

Taulukko 3.1. *Real AdaBoost* -algoritmi

1. Asetetaan havaintojen painotukset $w_i = \frac{1}{N}$, $i = 1, 2, \ldots, N$.
2. m käy läpi luvut $1, 2, \ldots, M$:
   (a) Sovitetaan luokittelija aineistoon, siten että saadaan laskettua luokan todennäköisyysestimaatti $p_m(x) = \hat{P}_w(y = 1|x) \in [0, 1]$, missä painotuksina käytetään edellisellä kierroksella laskettuja painoja w_i.
   (b) Asetetaan osapistemäärä additiiviselle mallille:
   $$f_m(x) \leftarrow \tfrac{1}{2} \log \tfrac{p_m(x)}{1-p_m(x)} \in \mathbb{R}.$$
   (c) Asetetaan uudet aineiston painotukset:
   $$w_i \leftarrow w_i \exp[-y_i f_m(x_i)],$$
   missä $i = 1, 2, \ldots, N$ ja painot normalisoidaan siten, että $\sum_{i=1}^{N} w_i = 1$.
3. Palautetaan luokittelija $G(x) = \text{sign}[\sum_{m=1}^{M} f_m(x)]$.

Taulukossa 3.1 on esitetty *Real AdaBoost* -algoritmi. Oletetaan, että käytettävä opetusaineisto on muotoa $L_i = (\boldsymbol{x}_i, y_i)$, missä $i = 1, 2, \ldots, N$. *Real AdaBoost* -algoritmiin syötetään selittävät muuttujat $\boldsymbol{x}_i$, selitettävä muuttuja $y_i \in \{0, 1\}$ sekä luokittelija $\hat{f}$, joka palauttaa arvon välillä $[0, 1]$. *Real AdaBoost* -algoritmin aluksi aineiston $L_i = (\boldsymbol{x}_i, y_i)$ havainnoille asetetaan yhtä suuret painot. Tämän jälkeen algoritmissa suoritetaan iterointeja M kertaa siten, että jokaisella kierroksella m aineistoon sovitetaan heikko luokittelija, jonka perusteella lasketaan luokkien todennäköisyysestimaatit. Saatujen estimaattien avulla päivitetään additiivinen malli ja lasketaan vielä aineistolle uudet painotukset seuraavia iterointikierroksia varten. Kun iterointeja on suoritettu M kertaa, palautetaan lopullinen luokittelija $G(x)$ enemmistöäänestyksen avulla (ks. Friedman, Hastie & Tibshirani, 2000).

3.4.2 *Gradient Boosting*

Gradient Boosting -algoritmi on yhdistelmäoppimisen menetelmiin kuuluva Friedmanin (1999) kehittämä pidemmälle kehitetty versio *AdaBoost*-algoritmista. *Stochastic Gradient Boosting* -algoritmi puolestaan pohjautuu *Gradient Boosting* -algoritmiin ja se on tarkoitettu erityisesti tilastollisten regressio-ongelmien ratkaisemi-

seen (Friedman 2002). Algoritmia voidaan kuitenkin käyttää myös luokittelutilanteissa pelkistämällä luokitteluongelmat regressioon sopivaan muotoon tappiofunktion avulla. Seuraavaksi tarkasteltavan *Gradient Boosting* -algoritmin teoreettinen osuus mukailee Friedmanin (1999) esitystä.

Kuten muissa *Boosting*-menetelmään perustuvissa algoritmeissa, myös *Gradient Boosting* -algoritmin estimointiongelma koostuu satunnaisesta selitettävästä muuttujasta y sekä satunnaisista selittävistä muuttujista $\boldsymbol{x} = \{x_1, x_2, \ldots, x_p\}$. Otos $L_i = (\boldsymbol{x}_i, y_i)$ $(i = 1, 2, \ldots, N)$ on puolestaan algoritmissa käytettävä opetusaineisto, josta tunnetaan arvot $(\boldsymbol{x}, y)$. Tällöin algoritmin ongelmana on löytää funktio $F^*(\boldsymbol{x})$, joka toimii kuvauksena vektorilta $\boldsymbol{x}$ pisteelle y siten, että se toimii kaikilla $(\boldsymbol{x}, y)$ arvoilla. Tämä saadaan yksinkertaisesti ratkaistua, kun jonkin määritellyn tappiofunktion $\Psi(F(\boldsymbol{x}), y)$ odotusarvo minimoidaan:

$$F^*(\boldsymbol{x}) = \underset{F(\boldsymbol{x})}{\arg\min}\, E_{y,\boldsymbol{x}}\, \Psi(F(\boldsymbol{x}), y). \tag{3.10}$$

Boosting-algoritmissa funktiota $F^*(\boldsymbol{x})$ approksimoidaan additiivisilla laajennuksilla

$$F(\boldsymbol{x}) = \sum_{m=0}^{M} \beta_m h(\boldsymbol{x}; \boldsymbol{a}_m), \tag{3.11}$$

missä perusoppijan (*base learner*) funktioiksi $h(\boldsymbol{x}, \boldsymbol{a})$ valitaan tavallisesti yksinkertaisia $\boldsymbol{x}$:n funktioita parametreilla $\boldsymbol{a} = \{a_1, a_2, \ldots\}$. Laajennuksen kertoimet $\{\beta_m\}_0^M$ ja parametrit $\{\boldsymbol{a}_m\}_0^M$ yhdistävät mallin opetusaineistoon etenevällä vaiheittaisella tavalla (*forward stagewise*). Mallin periaate on hyvin yksinkertainen: aluksi alustetaan funktio $F_0(\boldsymbol{x})$ asettamalla sille jokin lähtöarvo, jonka jälkeen lasketaan kaavat

$$(\beta_m, \boldsymbol{a}_m) = \underset{\beta, \boldsymbol{a}}{\arg\min} \sum_{i=1}^{N} \Psi(F_{m-1}(\boldsymbol{x}_i) + \beta h(\boldsymbol{x}_i; \boldsymbol{a}), y_i) \tag{3.12}$$

ja

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \beta_m h(\boldsymbol{x}; \boldsymbol{a}_m). \tag{3.13}$$

Gradient Boosting -algoritmissa ratkaistaan kaavan (3.12) ongelma likimääräisesti. Tämä tapahtuu mielivaltaisesti valitun derivoituvan tappiofunktion $\Psi(F(\boldsymbol{x}), y)$ ja kaksiaskeleisen algoritmin avulla. Algoritmin aluksi perusoppijasta $h(\boldsymbol{x}; \boldsymbol{a})$ lasketaan pienin neliösumma

$$\boldsymbol{a}_m = \underset{\boldsymbol{a}, \rho}{\arg\min} \sum_{i=1}^{N} [\tilde{y}_{im} - \rho h(\boldsymbol{x}_i; \boldsymbol{a})]^2, \tag{3.14}$$

missä $\tilde{y}_{im}$ on kutakin iterointikierrosta vastaavat 'pseudo'-residuaalit, jotka puolestaan saadaan laskettua kaavalla:

$$\tilde{y}_{im} = -\left[\frac{\partial \Psi(F(\boldsymbol{x}_i), y_i)}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x})}. \tag{3.15}$$

Tämän jälkeen funktiolle $h(\boldsymbol{x};\boldsymbol{a}_m)$ määritetään kertoimen β_m optimaalisin arvo kaavalla

$$\beta_m = \arg\min_{\beta} \sum_{i=1}^{N} \Psi(F_{m-1}(\boldsymbol{x}_i) + \beta h(\boldsymbol{x}_i;\boldsymbol{a}_m), y_i). \tag{3.16}$$

Taulukko 3.2. *Gradient Tree Boosting* -algoritmi

1. Asetetaan $F_0(\boldsymbol{x}) = \arg\min_{\gamma} \sum_{i=1}^{N} \Psi(\gamma, y_i)$, missä $\gamma \in \mathbb{R}$.
2. m käy läpi arvot $1, 2, \ldots, M$:
   (a) Lasketaan
   $$\tilde{y}_{lm} = -\left[\frac{\partial \Psi(F(\boldsymbol{x}_i), y_i)}{\partial F(\boldsymbol{x}_i)}\right]_{F(\boldsymbol{x})=F_{m-1}(\boldsymbol{x})}, \text{ missä } i = 1, 2, \ldots, N.$$
   (b) $\{R_{lm}\}_1^L$ = päätealueet L päätesolmun puusta $\{\boldsymbol{x}_i, \tilde{y}_{im}\}_1^N$
   (c) $\gamma_{lm} = \arg\min_{\gamma} \sum_{\boldsymbol{x}_i \in R_{lm}} \Psi(F_{m-1}(\boldsymbol{x}_i) + \gamma, y_i)$.
   (d) $F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \nu \cdot \sum_{l=1}^{L} \gamma_{lm} I(\boldsymbol{x} \in R_{lm})$.
3. Palautetaan $\hat{F}(\boldsymbol{x}) = F_M(\boldsymbol{x})$.

Gradient tree boosting -algoritmissa perusoppijana $h(\boldsymbol{x};\boldsymbol{a})$ on L päätesolmun regressiopuu. Jokaisella iteraatiokierroksella m regressiopuu jakaa vektorin $\boldsymbol{x}$ avaruuden L toisensa poissulkevaan päätealueeseen $\{R_{lm}\}_{l=1}^L$ ja ennustaa erillisen vakioarvon kullekin päätesolmulle

$$h(\boldsymbol{x}; \{R_{lm}\}_1^L) = \sum_{l=1}^{L} \bar{y}_{lm} I(\boldsymbol{x} \in R_{lm}), \tag{3.17}$$

missä $\bar{y}_{lm} = \text{mean}_{\boldsymbol{x}_i \in R_{lm}}(\tilde{y}_{im})$ eli kaavan (3.15) keskiarvo jokaisessa päätealueessa R_{lm}. Lisäksi $I(\boldsymbol{x} \in R_{lm})$ on indikaattorifunktio, joka saa arvon 1, kun $\boldsymbol{x}$ kuuluu päätealueeseen R_{lm} ja 0 muulloin. Perusoppijan parametrit ovat jakomuuttujat ja niitä vastaavat jakopisteet. Puu puolestaan määrittää sitä vastaavien päätealueiden $\{R_{lm}\}_1^L$ jaon iterointikierroksella m. Tämä ominaisuus mahdollistaa myös tutkielmassa myöhemmin käsiteltävän muuttujien tärkeysindeksin laskemisen, kun jaottelukriteerinä käytetään pienimmän neliösumman menetelmää.

Regressiopuissa kaava (3.16) voidaan ratkaista erikseen kullekin puun m päätesolmulle l. Koska kaavan (3.17) mukaan puu ennustaa vakioarvon $\bar{y}_{lm}$ kullekin päätealueelle R_{lm}, kaava (3.16) sievenee yksinkertaiseen paikalliseen estimaattiin perustuvaan kriteeriin Ψ

$$\gamma_{lm} = \arg\min_{\gamma} \sum_{\boldsymbol{x} \in R_{lm}} \Psi(F_{m-1}(\boldsymbol{x}_i) + \gamma, y_i). \tag{3.18}$$

Lopulta viimeisin approksimaatio $F_{m-1}(\boldsymbol{x})$ päivitetään erikseen jokaiselle vastaavalle päätealueelle

$$F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \nu \cdot \sum_{l=1}^{L} \gamma_{lm} I(\boldsymbol{x} \in R_{lm}). \tag{3.19}$$

Kutistusparametrilla $0 < \nu \leq 1$ kontrolloidaan proseduurin opetuksen suhdetta. Empiirisesti tutkimalla on löydetty, että pienet arvot ($\nu \leq 0.1$) johtavat pieneen yleistämisvirheeseen (Friedman 1999).

Tutkielman luokittelussa käytetty *Stochastic Gradient Boosting*-algoritmi pohjautuu taulukossa 3.2 esitettyyn *Gradient Tree Boosting*-algoritmiin. Merkittävimpänä erona on luokitteluongelmissa käytettävä tappiokriteeri sekä satunnainen permutointi.

Taulukko 3.3. *Stochastic Gradient Boosting* -algoritmi

1. Asetetaan $F_0(\boldsymbol{x}) = \arg\min_\gamma \sum_{i=1}^{N} \Psi(\gamma, y_i)$, missä $\gamma \in \mathbb{R}$.
2. m käy läpi arvot $1, 2, \ldots, M$:
   (a) $\{\pi(i)\}_1^N$ = satunnainen permutaatio $\{i\}_1^N$
   (b) Lasketaan
   $$\tilde{y}_{\pi(i)m} = -\left[\frac{\partial \Psi(F(\boldsymbol{x}_{\pi(i)}), y_{\pi(i)})}{\partial F(\boldsymbol{x}_{\pi(i)})}\right]_{F(\boldsymbol{x})=F_{m-1}(\boldsymbol{x})},$$
   missä $i = 1, 2, \ldots, \tilde{N}$.
   (c) $\{R_{lm}\}_1^L$ = päätealueet L päätesolmun puusta $\{\boldsymbol{x}_i, \tilde{y}_{im}\}_1^N$
   (d) $\gamma_{lm} = \arg\min_\gamma \sum_{\boldsymbol{x}_{\pi(i)} \in R_{lm}} \Psi(F_{m-1}(\boldsymbol{x}_{\pi(i)}) + \gamma, y_{\pi(i)})$
   (e) $F_m(\boldsymbol{x}) = F_{m-1}(\boldsymbol{x}) + \nu \cdot \sum_{l=1}^{L} \gamma_{lm} 1(\boldsymbol{x} \in R_{lm})$
3. Palautetaan $\hat{F}(\boldsymbol{x}) = F_M(\boldsymbol{x})$.

Oletetaan, että $\{y_i, \boldsymbol{x}_i\}_1^N$ on koko opetusaineisto ja $\{\pi(i)\}_1^N$ on satunnainen permutaatio kokonaisluvuista $\{1, 2, \ldots, N\}$. Oletetaan myös, että $\{y_{\pi(i)}, \boldsymbol{x}_{\pi(i)}\}_1^{\tilde{N}}$ ($\tilde{N} < N$) on satunnainen osaotos opetusaineistosta. Tutkielman aineiston kaltaisessa kaksiluokkaisessa luokittelutilanteessa oletetaan myös, että selitettävä muuttuja $y \in \{0, 1\}$ on Bernoulli-jakautunut, jolloin

$$P(y = 1|x) = \frac{1}{1 + e^{-2F(\boldsymbol{x})}}, \tag{3.20}$$

missä $F(\boldsymbol{x})$ saadaan kaavasta (3.11). Tällöin *Stochastic Gradient Boosting* -algoritmin tappiokriteerinä käytetään bernoullijakauman logaritmoidun uskottavuusfunktion devianssia -2:lla kerrottuna:

$$\Psi(y, \hat{F}) = 2\log(1 + e^{-2y\hat{F}}). \tag{3.21}$$

Näin *Stochastic Gradient Boosting* -algoritmi voidaan esittää taulukossa 3.3 esitetyn algoritmin mukaisesti. (Friedman 2002)

Kun asetetaan, että $\tilde{N} = N$, poistuu *Stochastic Gradient Boosting* -algoritmin satunnaisuus ja algoritmi (taulukko 3.3) palautuu *Gradient Tree Boosting* -algoritmiin (taulukko 3.2). Pienempi suhde $f = \tilde{N}/N$ antaa enemmän satunnaisia otoksia käyttöön jokaiselle iterointikierrokselle, jolloin saadaan enemmän satunnaisuutta koko algoritmiin. Otos on lähes samanlainen kuin bootstrap-otannalla poimittaessa, jos arvoksi asetetaan $f = 1/2$. On kuitenkin tärkeä huomata, että f:n pienentyessä, pienenee myös perusoppijan käytettävissä oleva datan määrä jokaisella iterointikierroksella. (Friedman 2002)

3.5 Satunnaismetsä

Satunnaismetsä (*Random Forest*) on oppimisyhdistelmä luokitteluun, joka koostuu useista luvussa 3.2.3 esitetyistä luokittelu- ja regressiopuista (*CART*). Breimanin (2001) kehittämä algoritmi sisältää päätöspuiden kaltaisia luokittelijoita, jotka rakennetaan aineistosta palauttaen poimituista bootstrap-otoksista. Bootstrap-otosten avulla rakennettujen päätöspuiden tarkoituksena on parantaa päätöspuun robustisuutta, jolloin saadaan parannettua luokittelun tarkkuutta.

Tarkastellaan seuraavaksi satunnaismetsää tarkemmin seuraten Hastie et el. (2011) kirjan lukua 15. Taulukossa 3.4 kuvatussa satunnaismetsä-algoritmissa metsän yksittäinen puu muodostetaan jokaisella iterointikierroksella $b = 1, 2, \ldots, B$ opetusaineiston $L_i = (\boldsymbol{x}_i, y_i)$ bootstrap-otoksesta. Oletetaan, että havaintojen lukumäärä alkuperäisessä opetusaineistossa L_i on N. Kullakin kierroksella opetusaineistosta poimitaan N_{tree} suuruinen bootstrap-otos $\boldsymbol{Z}^*$, missä $N_{tree} \leq N$. Poimitusta otoksesta kasvatetaan luokittelupuu T_b, jonka jokaisessa solmukohdassa (*node*) poimitaan m_{try} ($\leq M$) ennustajamuuttujan satunnaisotos M muuttujan joukosta. Poimitun otoksen perusteella valitaan paras jako, jonka mukaan havainnot ryhmitellään. Iterointia jatketaan, kunnes havaintojen ryhmittely havaintojen kesken on täydellinen, toisin sanoen niin kauan kuin jokaiselle oksalle jää vain samaan ryhmään kuuluvia havaintoja tai pienin solmun koko saavutetaan. Jokaiselle luokittelijalle voidaan myös laskea kussakin solmukohdassa tärkeysindeksi, joka kuvaa kyseessä olevan luokittelijan kykyä luokitella muuttujat.

Yleisesti satunnaismetsässä pyritään vähentämään ennustusfunktion varianssia bootstrap-otosten avulla. Kaavan (3.1) perusteella bootstrap-otosten ulkopuolelle jää noin 37% havainnoista. Näitä havaintoja kutsutaan *oob* -aineistoksi (*out of bag*), jota hyödynnetään luokitteluvirheen arvioinnissa. Tämän aineiston perusteella voidaan myös määrittää muuttujien tärkeysindeksi. Luokitteluvirhettä laskiessa havaintovektori $\boldsymbol{x}$ luokitellaan niillä puilla, joissa se on ollut *oob* -aineistossa eli niillä puilla, joiden muodostamiseen sitä ei ole käytetty. Tämä suoritetaan kaikille havainnoille ja lasketaan väärinluokiteltujen osuus. Satunnaismetsän puiden määrän kasvaessa *oob* -luokitteluvirhe e_{oob} lähestyy oikeaa luokitteluvirhettä (Breiman 2001). Tämä tarkoittaa sitä, että saadaan asymptoottisesti harhaton estimaattori luokitteluvirheelle, jolloin erillistä testiaineistoa tai esimerkiksi ristiinvalidointia ei tarvita. Menetelmässä pyritään valitsemaan kasvatettavien puiden lukumäärä siten, että menetelmän luokitteluvirhe on mahdollisimman pieni. Parhaaseen tulokseen päästään

valitsemalla käytettävien muuttujien lukumäärä, m_{try}, siten, että e_{oob} minimoituu. Lähes optimaaliset tulokset saavutetaan silloin, kun valitaan $m_{try} = \sqrt{M}$.

Taulukko 3.4. *Random Forest* -algoritmi regressiolle tai luokittelulle.

1. b käy läpi arvot $1, 2, \ldots, B$:
   (a) Poimitaan opetusaineistosta N_{tree} ($\leq N$) kokoinen bootstrap-otos $\mathbf{Z}^*$.
   (b) Kasvatetaan bootstrap-aineistosta satunnaismetsän puu T_b siten, että rekursiivisesti seurataan seuraavia vaiheita jokaisen puun solmun kohdalla. Lopetetaan kun pienin solmun koko n_{min} saavutetaan.
      i. Valitaan m_{try} muuttujaa satunnaisesti M muuttujan joukosta.
      ii. Valitaan joukon m_{try} jakomuuttuja ja -piste.
      iii. Jaetaan solmu kahteen tytärsolmuun.
2. Palautetaan puukokonaisuus $\{T_b\}_1^B$.

Tehdään ennuste uudelle pisteelle x:

Regressio: $\hat{f}_{rf}^B(x) = \frac{1}{B} \sum_{b=1}^{B} T_b(x)$.

Luokittelu: $\hat{C}_b(x)$ on b:nen *random forest* -puun luokkaennuste.
Tällöin $\hat{C}_{rf}^B(x)$ = enemmistöäänestys $\{\hat{C}_b(x)\}_1^B$.

3.6 Opetusaineiston tasapainon korjaaminen

Vaikka asiakaspoistuma on epäsuotuisa tapahtuma operaattoreille, se on tilastollisesti tarkasteltuna suhteellisen pientä. Vuonna 2012 suomalaisten operaattoreiden keskimääräinen poistumaprosentti kuukaudessa on vain 1,62%. Poistumaprosentin pienuuden ongelmana on, että tapahtuman harvinaisuus voi vaikeuttaa ennustamista. Tällöin on mahdollista, että asiakkaan luonteenomaiset vaihtuvuuden piirteet on vaikea havaita aineistosta ja toisaalta runsas pysyvien asiakkaiden osuus saattaa dominoida tuloksia tilastollisissa analyyseissa.

Lemmens & Croux (2006) mukaan yksikertainen ratkaisu ongelmaan on luoda tasapainotettu opetusaineisto, missä poistuneiden osuus on yhtä suuri kuin pysyvien osuus. Tämä ratkaisu on kuitenkin ongelmallinen, sillä tällöin luokittelija opetetaan tasapainotetulla opetusaineistolla, jolloin on mahdollista, että poistuneiden osuus yliestimoituu uusissa oikeissa havainnoissa. Ongelmaan on myös olemassa erilaisia luokittelijoita, mutta näitä ei ole vielä kehitetty uudempiin *Bagging* ja *Boosting* -menetelmiin.

Ensimmäisessä ratkaisussa liitetään painot havaintoihin siten, että opetusaineisto on tasapainossa poistuneiden ja pysyvien asiakkaiden välillä. Merkitään, että poistuvien asiakkaiden osuus on π_{churn}. Kun tiedetään poistuneiden osuus aineistossa, voidaan esimerkiksi estimoida empiiristä poistuneiden jakaumaa otoksessa. Merkitään myös, että N_{churn}^b on poistuneiden lukumäärä tasapainotetussa opetusaineistossa, missä N on tasapainotetun opetusaineiston koko. Tällöin yksi tapa painottaa

havainnot tasapainoisiksi on käyttää painotuksia

$$w_i^{churn} = \frac{\pi_{churn}}{N_{churn}^b} \text{ ja } w_i^{nochurn} = \frac{1 - \pi_{churn}}{N - N_{churn}^b} \tag{3.22}$$

poistuneille ja vastaavasti pysyville asiakkaille. Kaavalla (3.22) laskettujen painojen summa on aina yksi. Lisäksi poistuneiden havaintojen painojen summa saadaan liitettyä poistuneiden havaintojen alkuperäiseen suhteeseen kaavalla:

$$\pi_{churn} = \sum_{i=1}^{N_{churn}^b} w_i^{churn}. \tag{3.23}$$

Kun sovelletaan edellä mainittua painotuksen korjausta *Bagging*, *Stocastic Gradient Boosting* -algoritmeihin sekä satunnaismetsään, lasketaan painotettujen päätöspuiden sarja kuten tavallisestikin, mutta alkuperäiset opetusaineiston painotukset säilyvät kiinteinä kaikilla iterointikierroksilla. *Real AdaBoost* -algoritmissa annetaan alustaviksi painotuksiksi alkuperäisten painotuksien sijaan kaavalla (3.22) lasketut painot. Seuraavilla iterointikierroksilla käytetään normaalisti algoritmiin kuuluvia painotuksia. (Lemmens & Croux, 2006)

Toinen vaihtoehto tasapainotetun otoksen sijaan on määrittää nollasta poikkeava raja-arvo τ_B *Bagging* ja *Boosting*-algoritmeihin. Tätä menetelmää kutsutaan myös sieppauskorjaukseksi (katso lisää: Franses & Paap (2004), s.73-75). Arvo τ_B asetetaan siten, että ennustettu poistuneiden suhde $\hat{\pi}_{churn}$ opetusaineistossa on sama kuin alkuperäinen poistuneiden suhde π_{churn}. Tämä kriteeri saadaan aikaan *Bagging*-algoritmissa lajittelemalla ensin pisteet $\hat{f}_{bag}(x_i)$ suurimmasta pienimpään $\hat{f}_{bag}(x_1) \leq \hat{f}_{bag}(x_2) \leq \ldots \leq \hat{f}_{bag}(x_N)$ ja tämän jälkeen asetetaan τ_B kaavalla:

$$\tau_B = \hat{f}_{bag}(j), \text{ missä } j = N_{churn}^b. \tag{3.24}$$

Käsitellessä laajoja datoja on syytä muistaa, että absoluuttinen poistuneiden lukumäärä tarkasteltavassa opetusaineistossa ei koskaan ole merkityksetön. Vaikka poistuma onkin harvinainen tapahtuma, suhteellinen poiminta voi silti olla tehokasta, vaikka pienissä aineistoissa ositettu otanta olisi osuvampi valinta.

3.7 Muuttujien valinta

Tilastollisessa ennustamisessa pyritään ensimmäisenä löytämään ne muuttujat, jotka ovat mallin kannalta tärkeimmät. Tärkeimpien muuttujien avulla käytettävistä ennustemalleista on mahdollista saada yksinkertaisia ja tehokkaita. Tässä kappaleessa perehdytään muuttujanvalintaan, jota käytetään tilastollisissa oppimisyhdistelmäalgoritmeissa kuten esimerkiksi satunnaismetsässä.

Puuhun perustuvissa oppimisyhdistelmämenetelmissä naiivi tapa määrittää muuttujan tärkeys on laskea, kuinka monta kertaa kukin muuttuja valitaan yhdistelmän yksittäisiin puihin. Näissä menetelmissä on kuitenkin mahdollista laskea myös huomattavasti parempi mittasuure muuttujien tärkeydelle. Esimerkiksi satunnaismetsään Breiman (2001) johdattelee mittasuureeseen luokittelijamuuttujan tärkeydestä,

josta myöhemmin käytetään termiä tärkeysindeksi (*importance index*) (esim. Lunetta, Hayward, Segal & Eerdewegh, (2004)). Suure häiritsee selitettävän ja selittävien muuttujien välistä riippuvuutta sekä mittaa puiden äänien muutoksia vertaamalla uusia havaintoja alkuperäisiin havaintoihin. Käytännössä tämä toteutetaan permutoimalla muuttujien arvoja jokaisen puun *oob* -aineistossa. Jos muuttujalla on ennustekykyä, sitä esiintyy suurimmassa osassa puita ja sijaitsee kyseisten puiden juurten lähellä. Toisaalta on mahdollista, että havainto, jonka muuttujan arvo vaihtuu, saattaa ohjautua ohjata puun väärälle puolelle. Tällöin äänestys muuttuu oikeasta väärään luokkaan.

Satunnaismetsän tärkeysindeksi kuvaa luokittelumuuttujan kykyä erotella havainnot luokkiin. Indeksin perusteella muuttujien joukkoa pienennetään jokaisella analyysikierroksella siten, että pienimmät tärkeysindeksin arvot saaneet muuttujat tiputetaan pois seuraavilta analyysikierroksilta. Jakautumistarkkuuden parantuminen kerryttää luokittelumuuttujan tärkeysindeksiä mallin jokaisen puun kunkin solmun kohdalla. Kaikilta puilta luokittelumuuttujalle lasketuista tärkeysindeksiarvosta lasketaan lopuksi keskiarvo.

Seuraavaksi tarkastellaan tarkemmin tärkeysindeksiä. Olkoon jokaisella havainnolla i ennustajamuuttujien vektori $\boldsymbol{x}_i$ sekä todellinen luokka y_i. Olkoon puolestaan $v_j(\boldsymbol{x}_i)$ puun j antama ääni ja t_{ij} indikaattorimuuttuja, joka saa arvon 1 kun yksilö i on puun j *oob* -havainto ja muulloin t_{ij} saa arvon 0. Olkoon $\boldsymbol{x}^{(a,j)} = (\boldsymbol{x}_1^{(a,j)}, \boldsymbol{x}_2^{(a,j)}, \ldots, \boldsymbol{x}_N^{(a,j)})$ ennustajamuuttujien vektorien kokoelma, missä muuttujan a arvot on permutoitu satunnaisesti puun j *oob* -havainnolla ja olkoon $\boldsymbol{x}^{(a)}$ kokoelmien $\boldsymbol{x}^{(a,j)}$ kokoelma, missä N on otoksessa olevien havaintojen lukumäärä. Olkoon myös $I(C)$ indikaattorifunktio, joka saa arvon 1 kun C on tosi ja 0 kun C on epätosi. Tällöin tärkeysindeksi on keskiarvo metsän kaikista puista ja voidaan laskea kaavalla

$$I_T(a) = \frac{1}{T}\sum_{j=1}^{T}\frac{1}{N_j}\sum_{i=1}^{N}[I(v_j(\boldsymbol{x}_i) = y_i) - I(v_j(\boldsymbol{x}^{(a,j)}) = y_i)]t_{ij}, \tag{3.25}$$

missä N_j kuvaa puun j *oob* -havaintojen lukumäärää ja T puiden kokonaislukumäärää metsässä. (Lunetta et al. 2004)

3.8 Mallien arviointikriteerit

Tutkielmassa vertaillaan eri tilastollisten oppimismenetelmien sovelluksia asiakasvaihtuvuusaineistoon. Kuten tilastotieteessä yleensä, on myös asiakasvaihtuvuuden ennustamisessa tärkeää valita oikeanlaiset mallinvalinnan kriteerit, jotta saataisiin luotettavasti vertailtua erilaisia menetelmiä keskenään.

3.8.1 Ennustevirhe

Tilastotieteessä yksi perinteisimmistä tavoista vertailla tilastollisia malleja on tarkastella mallien ennustevirheitä ja tehdä päätelmiä niiden perusteella. Yleensä ennustevirhettä laskettaessa alkuperäinen aineisto jaetaan kahteen osaan: opetusaineistoon

ja testiaineistoon. Opetusaineiston avulla rakennetaan luokittelusääntö, jonka toimivuutta voidaan testata testiaineiston avulla. Tavoitteena tilastollisten mallien arvioinnissa on siis tietää estimoidun mallin $\hat{f}$ odotettu testivirhe.

Ennustevirhettä laskiessa on syytä huomata, että esimerkiksi opetusaineistosta laskettu opetusvirhe ei ole hyvä estimaatti ennustevirheelle, koska opetusvirhe on määritelty siten, että ennustevirheen arviointiin käytetään samoja havaintoja, joilla estimoitu malli on rakennettu. Tästä seuraa, että mallin kompleksisuuden kasvaessa ennustevirhe laskee. Tällöin on mahdollista, että ennustevirheen arvoksi saadaan $\overline{\text{err}} = 0$, jolloin estimoitu malli ylisovittuu aineistoon. Tämän seurauksena estimoidun mallin opetusvirhettä ei tyypillisesti voida yleistää kovin hyvin.

Hastie et al. (2011) määrittelevät luvussa 7 opetus- ja testiaineistoon jaetun aineiston ennustevirheet seuraavasti. Oletetaan, että alkuperäinen aineisto koostuu vastemuuttujasta y ja selittävistä muuttujista $\boldsymbol{x} = \{x_1, x_2, \ldots, x_p\}$. Otos $\tau = (\boldsymbol{x}_i, y_i)$ $(i = 1, 2, \ldots, N)$ on puolestaan käytettävä opetusaineisto, josta tunnetaan arvot $(\boldsymbol{x}, y)$. Merkitään, että $\hat{f}(\boldsymbol{x})$ on opetusaineistoon τ sovitettu ennustefunktio, jolloin tappiofunktiolla $L(y, \hat{f}(\boldsymbol{x}))$ mitataan virhettä y:n ja $\hat{f}(\boldsymbol{x})$:n välillä. Luokittelumenetelmien tilanteessa tappiofunktioksi asetetaan

$$L(y, \hat{f}(\boldsymbol{x})) = \begin{cases} 0, & \text{jos } \hat{f}(\boldsymbol{x}) = y \\ 1, & \text{jos } \hat{f}(\boldsymbol{x}) \neq y \end{cases} . \tag{3.26}$$

Testivirhe on puolestaan ennustevirhe, joka voidaan laskea mallista riippumattoman testiaineiston avulla käyttäen kaavaa:

$$\text{Err}_\tau = E[L(y, \hat{f}(\boldsymbol{x})) \mid \tau], \tag{3.27}$$

missä $(\boldsymbol{x}, y)$ on poimittu satunnaisesti alkuperäisestä aineistosta. Luokittelumenetelmien tilanteessa ennustevirhe on tappiofunktiota (3.26) käytettäessä väärinluokiteltujen havaintojen osuus. Vastaavasti odotettu testivirhe (tai odotettu ennustevirhe) saadaan kaavalla

$$\text{Err} = E[L(y, \hat{f}(\boldsymbol{x}))] = E[\text{Err}_\tau]. \tag{3.28}$$

Tällöin voidaan laskea opetusvirhe, joka on opetusaineiston keskimääräinen tappio:

$$\overline{\text{err}} = \frac{1}{N} \sum_{i=1}^{N} L(y_i, \hat{f}(\boldsymbol{x_i})), \tag{3.29}$$

missä N on opetusaineiston havaintojen lukumäärä. Jos taas käytetään kaavan (3.26) tappiofunktiota, opetusvirhe saadaan laskemalla väärinluokiteltujen havaintojen osuus.

Usein yhdistelmäoppimismenetelmissä aineisto jaetaan opetus-, validointi- ja testiaineistoon. Aineiston jakaminen kolmeen osaan ei kuitenkaan aina ole mahdollista. Siksi on kehitetty useita erilaisia menetelmiä, joilla voidaan arvioida mallin ennustevirhettä. Yksi toimiva ja tutkielman algoritmeissa käytetty menetelmä on bootstrap-menetelmä, jossa ennustevirhettä arvioidaan bootstrap-otosten avulla. Merkitään opetusaineistoksi $\boldsymbol{Z} = (Z_1, Z_2, \ldots, Z_N)$, missä $Z_i = (\boldsymbol{x_i}, y_i)$. Poimitaan

tästä opetusaineistosta palauttaen N alkion otos B kertaa (yleensä $B = 100$), jolloin saadaan B bootstrap-aineistoa $Z^* = (Z^{*1}, Z^{*2}, \ldots, Z^{*B})$. Bootstrap-menetelmällä voidaan arvioida esimerkiksi mallin luokitteluvirhettä laskemalla jokaiselle bootstrap-otokselle Z^{*b} luokittelusääntö. Jos $\hat{f}^{*b}(x_i)$ on piirrevektorin $\boldsymbol{x}_i$ ennustama luokka b:nnessä bootstrap-otoksessa, niin luokittelusäännölle lasketaan luokitteluvirhe alkuperäisestä opetusaineistosta $\boldsymbol{Z}$ kaavalla:

$$\widehat{\mathrm{Err}}_{\mathrm{boot}} = \frac{1}{B}\frac{1}{N}\sum_{b=1}^{B}\sum_{i=1}^{N} L(y_i, \hat{f}^{*b}(x_i)). \tag{3.30}$$

Kuitenkin on helppo havaita, että $\widehat{\mathrm{Err}}_{\mathrm{boot}}$ ei ole yleisesti hyvä estimaatti ennustevirheelle, koska opetus- ja testiaineistossa on samoja havaintoja. Tästä syystä estimaatti ylisovittuu helposti.

Ylisovittamiseen liittyvään ongelmaan löytyy ratkaisu bootstrap-prosessilla saadusta *oob* -aineistosta eli niistä havainnoista, jotka jäävät bootstrap-otosten ulkopuolelle. Poimittaessa bootstrap-otoksia, on kaavan (3.1) perusteella jokaisella havainnolla noin 63% todennäköisyys tulla valituksi otokseen. Näin ollen otoksen ulkopuolelle jääviä 37% aineistosta voidaan hyödyntää bootstrap-testiaineistona. Sen avulla voidaan laskea satunnaismetsä-algoritmissa käytettyä *leave-one-out* -estimaatia ennustevirheelle kaavalla:

$$\widehat{\mathrm{Err}}^{(1)} = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{|C^{-i}|}\sum_{b\in C^{-i}} L(y_i, \hat{f}^{*b}(x_i)), \tag{3.31}$$

missä C^{-i} on kaikkien niiden bootstrap-otosten joukko, jotka eivät sisällä havaintoa i. $|C^{-i}|$ on puolestaan kaikkien edellä mainittujen havaintojen lukumäärä.

3.8.2 Ristiinvalidointi

Mahdollisesti yksinkertaisin ja yleisimmin käytetty menetelmä ennustevirheen arvioinnissa on ristiinvalidointi (*cross validation*), jota tässä kappaleessa käsitellään Hastie et al. (2011) kirjan luvun 7.10 pohjalta. Menetelmässä aineisto jaetaan K osaan siten, että jokaisessa osassa on n kappaletta havaintoja. Jokaiselle osalle luokittelusääntö rakennetaan $K - 1$ osalla, jolloin luokitteluvirhe voidaan laskea sillä osalla, joka on jätetty pois luokittelusäännöstä. Tämä tehdään K kertaa ja yhdistetään K ennustevirhettä laskemalla niiden keskiarvo. Tyypillisesti luvuksi K valitaan 5 tai 10. Tilanne, jossa $K = N$, tunnetaan *leave-one-out* -ristiinvalidointina.

Olkoon $\kappa : \{1, 2, \ldots, N\} \to \{1, 2, \ldots, K\}$ indeksifunktio, joka indikoi paikan, mihin havainto i satunnaisesti arvotaan. Merkitään, että $\hat{f}^{-k}(x)$ on sovitettu malli kun k:s osa aineistosta on siirretty syrjään. Tällöin ristiinvalidoitu estimaatti ennustevirheelle on

$$\mathrm{CV}(\hat{f}) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \hat{f}^{-\kappa(i)}(x_i)). \tag{3.32}$$

Aineiston luokittelijan ja testivirheen kannalta aineiston tarkka jako opetus- ja testiaineistoon on ongelmallinen, sillä yleisesti ottaen mitä suurempi opetusaineisto on, sitä parempi luokittelija saadaan. Toisaalta tarkastellessa ennustevirhettä saavutetaan sitä parempia ennustevirheen arvoja, mitä suurempi testiaineisto on. Opetusaineiston koko voi vaikuttaa myös ennustevirheen estimaatin harhaan, jos aineisto on liian riippuvainen siitä. Myös yllä esitetyissä ennustevirheen estimointimenetelmissä aineiston koolla on merkitystä. Esimerkiksi K-kertaisessa ristiinvalidoinnissa, ennustevirheellä voi olla pieni varianssi, mutta silti estimaatti voi olla harhallinen, jos aineisto on liian pieni. Bootstrap-menetelmässä puolestaan otokseen päätyy aina hieman yli puolet havainnoista, joten ennustevirheen $\widehat{\overline{\text{Err}}}^{(1)}$ harha käyttäytyy vastaavasti kuin kaksinkertaisessa ristiinvalidoinnissa. Bootstrap-menetelmässä opetusaineiston koosta johtuvaa harhaa voidaan myös korjata estimaattorilla:

$$\widehat{\overline{\text{Err}}}^{(0.632)} = 0.368 * \overline{\overline{\text{err}}} + 0.632 * \widehat{\overline{\text{Err}}}^{(1)}. \tag{3.33}$$

Estimaattorin avulla saadaan pienennettyä ylöspäin harhaisen *leave-one-out* -bootstrap-estimaatin arvoa kohti opetusvirhettä.

3.8.3 Parhaan kymmenyksen noste

Yrityksen näkökulmasta kaikkein mielenkiintoisimpia tapauksia ovat ne asiakkaat, jotka ovat todennäköisimmin poistumassa seuraavana. Parhaan kymmenyksen nosteen avulla saadaan kohdistettua kriittisimpien asiakkaiden ryhmä, joiden poistumariski on kaikkein suurin. Ryhmä koostuu 10 prosentista asiakkaita, joilla on suurin poistuvuutta ennustava pisteluku. Yrityksen kannalta nämä asiakkaat edustavat mahdollisesti ihanteellisinta ryhmää, johon voi kohdistaa uudelleenmarkkinointia. Määritellään

$$\text{Parhaan kymmenyksen noste} = \frac{\hat{\pi}_{10\%}}{\hat{\pi}}, \tag{3.34}$$

missä $\pi_{10\%}$ on poistuneiden osuus riskialttiissa segmentissä ja $\hat{\pi}$ on poistuneiden osuus koko aineistossa. Tällöin luokittelija on sitä parempi, mitä suurempi on parhaan kymmenyksen noste. Tällä suureella on myös mahdollista tarkistaa ovatko kohdesegmentissä riskialttiit asiakkaat todella sisältyneet poistuneihin asiakkaisiin. (Lemmens & Croux, 2006)

4 Aineiston analyysi

Tutkielman tarkoituksena on löytää parhaat muuttujat sekä parhaat tilastolliset mallinnusmenetelmät, joilla selitetään asiakasvaihtuvuutta. Operaattorin kannalta on mielekästä tietää, mitkä seikat johtavat poistumiseen. Niiden avulla operaattori voi kehittää palveluita ja parantaa asiakassuhteiden hoitamista. Toisaalta myös mallinvalinta on tärkeää, sillä ennustaessa seuraavaksi poistuvia asiakkaita olisi hyvä, että mallin virheellinen luokittelu olisi mahdollisimman pientä.

Mallinnuksessa käytetään R-ohjelmistoa (R Core Team 2013), ja mallit tehdään sekä painottamattomalle että painotetulle aineistolle. Luonnollisesti painottamattomalla aineistolla saavutetaan pienempiä luokitteluvirheitä kuin painotetulla aineistolla, koska vaihtuvien asiakkaiden lukumäärä painottamattomassa aineistossa on hyvin vähäistä. Vaihtuvuuden vähäisyys tuo kuitenkin ongelmia siinä, että kaikista poistuneista asiakkaista havaitaan vain pieni osa, vaikka mallin luokitteluvirhe on hyvin pieni. Hyvin pieneen luokitteluvirheeseen päästään myös sillä, että luokitellaan kaikki havainnot pysyviksi asiakkaiksi, jolloin luokitteluvirhe aineistossa on vain 1,523% eli poistumaprosentin verran.

Luokittelumenetelmiä käytettäessä painottamattoman aineiston ongelma on myös se, että luokittelija oppii havainnot, jotka pysyvät asiakkaina. Tällöin virheelliset luokittelut kohdistuvat erityisesti niihin asiakkaisiin, jotka ovat vaihtumassa. Painotetulla aineistolla pyritään pääsemään eroon tästä ongelmasta. Optimaalisin tilanne olisi silloin, kun luokittelija oppisi tunnistamaan vaihtuvan asiakkaan ja pysyvän asiakkaan yhtä hyvin.

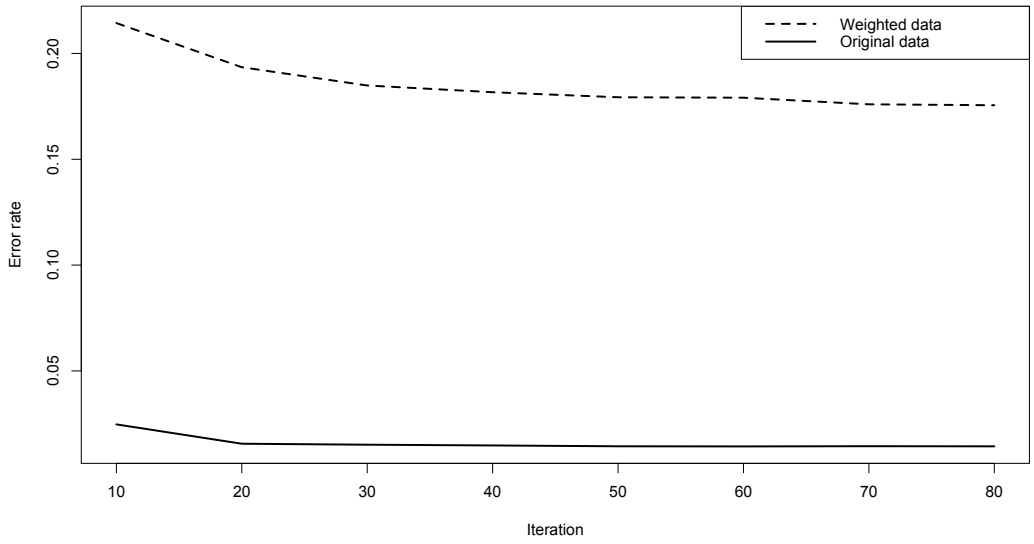
Molemmat tutkielmassa käytetyt aineistot jaettiin samalla tavalla kahteen osaan: opetus- ja testiaineistoon. Opetusaineiston havainnot poimittiin satunnaisesti alkuperäisestä aineistosta ja sisälsivät 60% aineiston havainnoista. Testiaineistoon jäi puolestaan ne havainnot, jota eivät tulleet valituiksi opetusaineistoon. Opetusaineistot ja testiaineistot muodostettiin vain kerran ja niitä käytettiin jokaisen mallin kohdalla, jotta testi- eli luokitteluvirhe olisi mahdollisimman luotettava eri mallien välillä. Myöhemmin eri menetelmien luokitteluvirheen suppenemista käsittelevissä kuvioissa on syytä huomata, että kuvioiden luokitteluvirheet on laskettu opetusaineistosta, jolla menetelmät on opetettu. Tällöin kuvaajista näkee hyvin, miten luokitteluvirhe pienenee iterointikierrosten edetessä. Menetelmien tehokkuutta vertaillessa arvioidaan kuitenkin vain testiaineistosta saatuja luokitteluvirheitä.

Aineiston analyysissä edetään siten, että ensin käsitellään jokainen menetelmä niin painotetulla kuin painottamattomalla aineistolla. Lisäksi tutkitaan menetelmien valitsemia tärkeimpiä muuttujia kummallakin aineistolla. Tämän jälkeen vertaillaan malleja keskenään ja valitaan niistä parhaat painottamattomalle ja painotetulle aineistolle. Lopuksi vertaillaan vielä painotetun ja painottamattoman aineiston paremmuutta parhaan kymmenyksen nosteen avulla. Eräisiin kuvioihin on vielä lisätty vertailuarvoksi päätöspuiden luokitteluvirhe, jotta nähdään, millainen vaikutus tilastollisella yhdistelmäoppimismenetelmillä on luokitteluvirheen suuruuteen.

4.1 *Bagging*

Mallinnettaessa aineistoja *Bagging*-menetelmällä, poimitaan aineistoista bootstrap-otoksia palauttaen ja kasvatetaan otoksista päätöspuita. Saaduista päätöspuista äänestetään lopullinen *Bagging*-puu enemmistöäänestyksellä. Näin saadaan vähennettyä yhden päätöspuun tuomaa virhettä.

Tutkielman *Bagging*-algoritmin mallinnuksessa käytetään R-ohjelmiston 'ipred'-pakkausta (Peters, Hothorn, Ripley, Therneau & Atkinson, 2013). *Bagging*-menetelmässä iterointeja suoritetaan 80 kertaa. Lisäksi menetelmää toistettiin 50 kertaa painottamattomalle ja painotetulle aineistolle, jotta saataisiin luotettava käsitys mallin tarkkuudesta. Saaduista malleista laskettiin ennustevirheet (err), joiden keskiarvot ($\overline{\text{err}}$) toimivat estimaattina mallin ennustevirheelle painottamattoman ja painotetun aineiston tapauksissa. Painottamattoman aineiston keskimääräinen ennustevirhe oli $\overline{\text{err}} = 0.0144$ ja painotetun aineiston keskimääräinen ennustevirhe oli $\overline{\text{err}} = 0.1712$. Kuviossa 4.1 nähdään, miten keskimääräinen ennustevirhe pienenee iterointikierros-



Kuvio 4.1. Painottamattoman- ja painotetun aineiston luokitteluvirhe *Bagging*-algoritmilla mallinnettuna, kun luokitteluvirhe on laskettu opetusaineistosta.

ten lisääntyessä. Painotetulla aineistolla minimaalinen virheiden osuus aineistossa saavutetaan 80 kierroksen paikkeilla, kun taas painottamattomassa aineistossa ennustevirhe ei pienene enää 50 kierroksen jälkeen.

4.2 *Real AdaBoost*

Binäärisiin luokittelu- ja regressio-ongelmiin tarkoitettu *Real AdaBoost* -algoritmi perustuu heikkoihin luokittelijoihin, joita sovelletaan perättäin aineistoon siten, et-

tä jokaisen luokituksen välissä aineisto painotetaan uudelleen edellisen luokittelijan perusteella. Luokittelijoina voidaan käyttää esimerkiksi luokittelu- ja regessiopuita, joista tutkielmassa aineistoon sovellettiin luokittelupuita. Mallin lopullinen luokittelija rakennetaan saaduista luokittelijoista enemmistöäänestyksellä. Näin saadaan pienennettyä yksittäisen luokittelijan luokitteluvirhettä.

Real AdaBoost-algoritmin tarkasteluun tutkielmassa käytetään R-ohjelmiston 'ada'-pakkausta. Tällä pakkauksella voidaan mallintaa erilaisia *AdaBoost* -algoritmeja kuten myös tutkielmassa tutkittua *Real AdaBoost* -algoritmia. Pakkaus sisältää myös tärkeysindeksin, jonka avulla voidaan arvioidan muuttujien tärkeyttä. (Clup, Johnson & Michailidis, 2012)

Samoin kuin muita algoritmeja soveltaessa, myös *Real AdaBoost* -algoritmin luokittelija opetetaan opetusaineistolla ja luokittelijan luokitteluvirhettä arvioidaan testiaineiston avulla. Jotta algoritmin luokitteluvirheelle saataisiin luotettava estimaatti, toistetaan menetelmää 50 kertaa. Jokaisesta toistosta lasketaan luokitteluvirhe, jolloin todellinen estimaatti mallin luokitteluvirheelle saadaan luokitteluvirheiden keskiarvosta $\overline{\text{err}}$. Painottamattomassa aineistossa luokitteluvirhe oli $\overline{\text{err}} = 0.0136480$ ja painotetussa aineistossa puolestaan $\overline{\text{err}} = 0.177450$.

Taulukko 4.1. Painottamattoman ja painotetun aineiston tärkeimmät muuttujat *Real AdaBoost* -algoritmilla mallinnettuna. Tarkemmat kuvaukset taulukossa mainituista muuttujista löytyvät liitteestä.

Tärkeys	Painottamaton aineisto	Pisteet	Painotettu aineisto	Pisteet
1	sopimuksen_ika_luokiteltu	0.11696	sopimuksen_ika_luokiteltu	0.06397
2	POSTAL_CODE_AREA	0.08814	n_12kk	0.05969
3	jatkoja_luokiteltu	0.08808	puhe_12kk	0.05550
4	jatkoja	0.08807	n_3kk	0.05513
5	OS	0.06432	puhe_6kk	0.05508
6	sex	0.06096	puhe_3kk	0.05387
7	paate_ika	0.04968	data_6kk	0.05387
8	ika_1	0.04396	n_6kk	0.05189
9	data_1to3_luok	0.04310	muutos_puhe	0.05184
10	n_1to3_luok	0.04310	data_3kk	0.05176

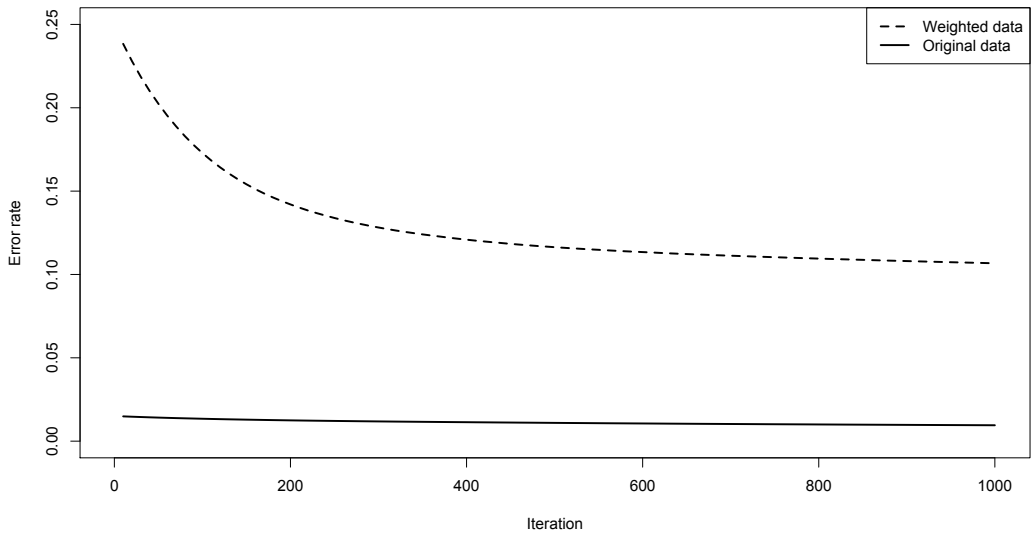
Taulukosta 4.1 on esitetty kymmenen merkitsevintä muuttujaa painottamattomassa ja painotetussa aineistossa, kun aineistojen muuttujien tärkeysindeksit on mallinnettu *Real AdaBoost* -algoritmilla. Muuttuja *sopimuksen_ika_luokiteltu* on selkeästi tärkein muuttuja asiakasvaihtuvuuden kannalta molemmissa aineistossa. Tämän jälkeen muuttujat eroavat toisistaan täysin. Painotetussa aineistossa mallin mukaan puhelimen käyttöön liittyvät muuttujat selittävät vaihtuvuutta parhaiten kun taas painottamattomassa aineistossa sopimuksen ja liittymän omistajan taustamuuttujat nousevat tärkeimmiksi tekijöiksi.

4.3 *Stochastic Gradient Boosting*

Stochastic Gradient Boosting -algoritmi on erityisesti regressio-ongelmiin tarkoitettu tilastollinen koneoppimisalgoritmi. Kuitenkin menetelmää voidaan käyttää myös

luokitteluongelmissa pelkistämällä tappiofunktio regressioon sopivaan muotoon.

Bagging-algoritmin mallinnuksessa käytetään R-ohjelmiston 'gbm'-pakkausta, joka sisältää *Stochastic Gradient Boosting* -algoritmin lisäksi myös muuttujan valintaan tarvittavan tärkeysindeksin (Ridgeway 2013). Menetelmä on varsin hidas, mutta sillä on kuitenkin mahdollista saavuttaa hyvin pieniä luokitteluvirheitä. Jotta aineiston mallinnuksessa päästäisiin parhaisiin mahdollisiin tuloksiin, suoritetaan 1000 iterointikierrosta painottamattomalle ja painotetulle aineistolle. Mallinnus toistetaan 10 kertaa, jolloin luokitteluvirheen estimaatiksi, $\overline{\text{err}}$, saadaan jokaisen kierroksen luokitteluvirheen keskiarvo. Näin saadaan luokitteluvirheeksi painottamattomalle aineistolle $\overline{\text{err}} = 0.0134$ ja painotetulle aineistolle $\overline{\text{err}} = 0.1685$.



Kuvio 4.2. Painottamattomaan ja painotettuun aineistoon sovellettujen mallien luokitteluvirheet eri iterointikierroksilla. Virheet on laskettu opetusaineistosta ja mallinnukseen on käytetty *Stochastic Gradient Boosting* -algoritmia.

Kuviossa 4.2 on esitetty *Stochastic Gradient Boosting* -algoritmin opetusaineiston luokitteluvirheet eri iterointikierroksilla. Siitä nähdään, kuinka opetusaineiston luokitteluvirhe pienenee iterointikierrosten kasvaessa. Huomattavaa on, että luokitteluvirhe pienenee hyvin hitaasti niin painottamattomassa kun painotetussa aineistossa. Laajoissa aineistoissa iterointien lukumäärä korostuu, sillä jokainen kierros vaatii paljon laskentatehoa. Toisaalta kyseisen algoritmin avulla on mahdollista saavuttaa hyvä tarkkuus kummankin aineistojen kaltaisissa tilanteissa, jos mallintamiseen on riittävästi aikaa käytettävissä.

Taulukossa 4.2 on esitettynä R-ohjelmiston 'gbm'-pakkauksen *Stochastic Gradient Boosting* -algoritmiin sisällytetty muuttujien valinnan kymmenen parasta muuttujaa sekä painottamattoman että painotetun aineiston tilanteessa. Aineiston painottaminen tuottaa eroja tärkeimmissä muuttujissa, myös *Stochastic Gradient Boosting* -algoritmilla mallintaessa. Molemmista aineistoista kymmenen parhaan joukkoon

Taulukko 4.2. Painottamattoman ja painotetun aineiston tärkeimmät muuttujat, kun asiakasvaihtuvuutta mallinnetaan *Stochastic Gradient Boosting* -algoritmilla. Tarkemmat kuvaukset taulukossa mainituista muuttujista löytyvät liitteestä.

Tärkeys	Painottamaton aineisto	Pisteet	Painotettu aineisto	Pisteet
1	ika_1	20.22154	jatkoja_luokiteltu	37.65169
2	POSTAL_CODE_AREA	15.24355	sopimuksen_ika	11.43487
3	sopimuksen_ika	12.38766	n_3kk	11.00505
4	PRODUCT_GRP_NAME	7.02105	paate_ika	7.84106
5	paate_ika	5.71307	jatkoja	4.78484
6	lasku	3.58179	ika_1	3.64236
7	jatkoja_luokiteltu	3.12957	POSTAL_CODE_AREA	3.04332
8	OS	2.77174	puhe_3kk	2.98701
9	jatkoja	1.81690	muutos_n	2.61907
10	data_12kk	1.62391	PRODUCT_GRP_NAME	2.58913

valikoituu kuitenkin seitsemän samaa muuttujaa. Näin voitaisiin ajatella, että myös painottamattomalla aineistolla saadaan hyviä tuloksia asiakasvaihtuvuutta ennustettaessa.

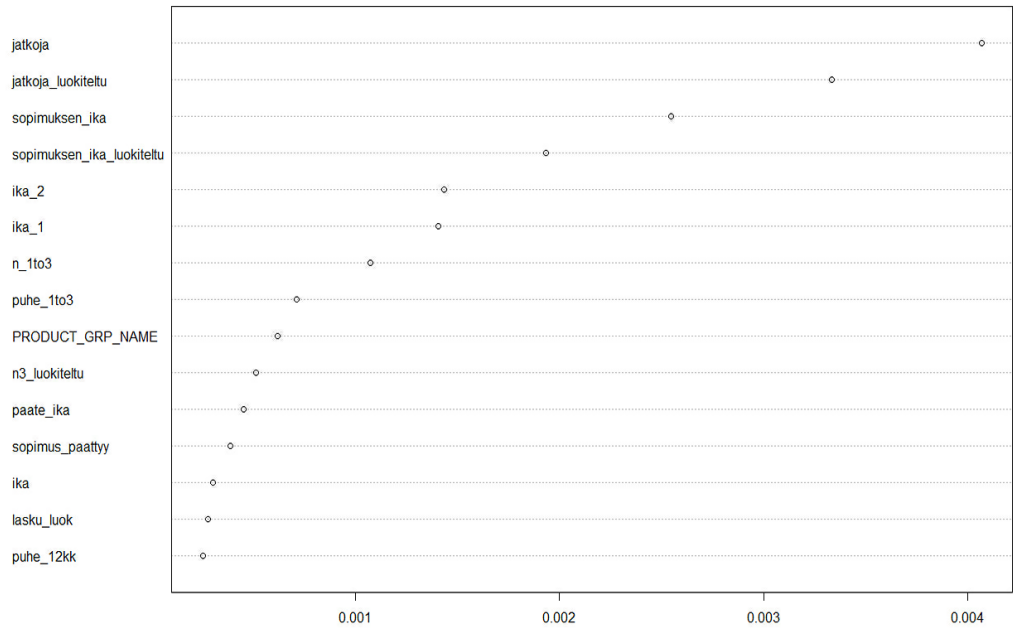
4.4 Satunnaismetsä

Satunnaismetsä muodostuu useasta satunnaisesti rakennetusta päätöspuusta. Menetelmä ei tarvitse erillistä testiaineistoa, sillä testiaineisto saadaan havainnoista, jotka eivät tule valituksi bootstrap-otannalla opetusaineistoon.

Tutkielmassa satunnaismetsä-algoritmin mallinnukseen käytetään R-ohjelmiston 'randomForest'-pakkausta (Breiman, Cutler, Liaw & Wiener, 2010). Malliin syötetään koko opetusaineisto, josta simuloidaan 50 metsää. Jokainen metsä koostuu 70 puusta. Yksittäinen puu puolestaan muodostetaan 8 muuttujasta, jotka on satunnaisesti valittu 54 selittävän muuttujan joukosta. Metsien mallinnusta kokeiltiin eri muuttujien lukumäärällä ja 8 muuttujalla saavutettiin pienin luokitteluvirhe. Lukumäärä onkin hyvin lähellä luvussa 3.5 mainittua metsän rakentamiseen käytettävien muuttujien optimaalista määrää.

Satunnaismetsän luokitteluvirheelle saadaan estimaatti *out-of-bag* -virheiden keskiarvosta. Kun molemmista aineistosta simuloidaan 50 metsää, saadaan luokitteluvirheen estimaatiksi painottamattomalle aineistolle $\overline{e_{oob}} = 0.01355$ ja painotetulle aineistolle $\overline{e_{oob}} = 0.1702$. Otokseen kuulumattomien havaintojen eli *oob* -havaintojen avulla laskettu e_{oob} vastaa opetusaineiston kokoisen ja riippumattoman testiaineiston luokitteluvirhettä.

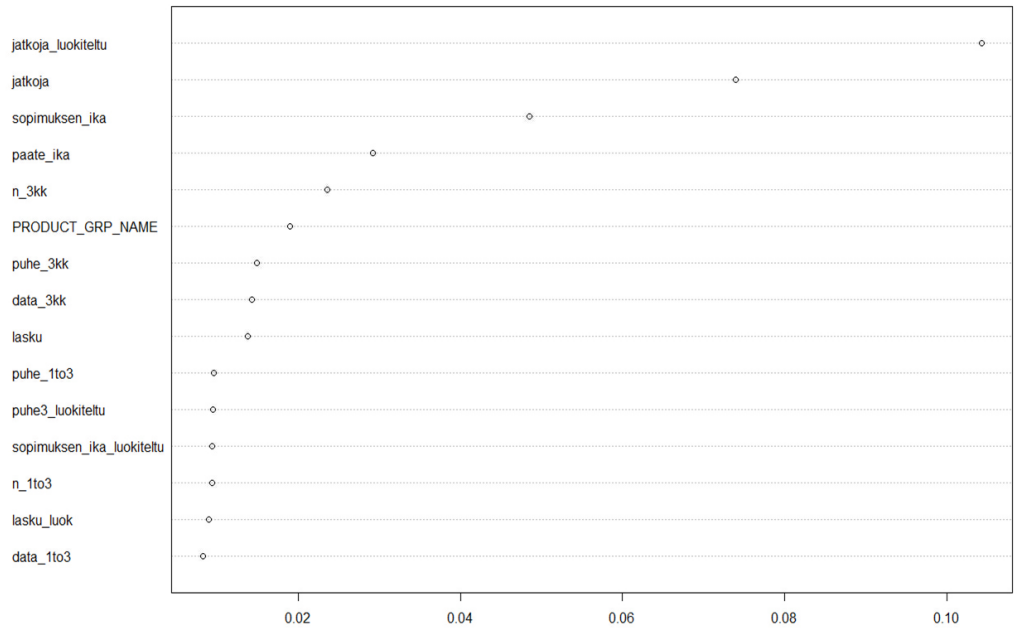
Satunnaismetsä-algoritmilla voidaan myös arvioida muuttujien tärkeyttä. Kuviossa 4.3 nähdään asiakasvaihtuvuuden kannalta 15 tärkeintä muuttujaa, kun painottamattoman aineiston muuttujien tärkeyttä on arvioitu satunnaismetsän tärkeysindeksillä. Tärkeistä muuttujista voidaan päätellä, että perättäisten liittymien lukumäärä (*jatkoja*) on selkeästi tärkein muuttuja. Lisäksi *jatkoja*-muuttujasta on tehty luokittelumuuttuja, *jatkoja_luokiteltu*, jossa on neljä eri luokkaa. Myös tämä muut-



Kuvio 4.3. Painottamattomassa aineistossa asiakasvaihtuvuuden kannalta tärkeimmät muuttujat mallinnettuna satunnaismetsä-algoritmilla, jossa vaaka-akselilla kuvataan muuttujan tärkeysindeksin saamaa arvoa. Tarkemmat kuvaukset kuviossa mainituista muuttujista löytyvät liitteestä.

tuja on tärkeä tärkeysindeksillä arvioituna. Kolmantena asiakasvaihtuvuuden kannalta tärkeänä muuttujana on aikaa viimeisimmästä sopimuksesta kuvaava muuttuja *sopimuksen_ika*. Kuviossa 4.4 puolestaan näkyy painotetun aineiston 15 tärkeintä muuttujaa, kun muuttujien tärkeyttä arvioidaan satunnaismetsän tärkeysindeksillä. Kuten painottamattomassa aineistossa, myös painotetussa aineistossa vaihtuvuuden kannalta tärkeimmät muuttujat ovat luokiteltu perättäisten liittymien lukumäärä (*jatkoja_luokiteltu*), jatkojen lukumäärä (*jatkoja*) ja aika viimeisimmästä sopimuksesta (*sopimuksen_ika*). Tämän jälkeen seuraavat muuttujat poikkeavat painottamattoman ja painotetun aineiston välillä. Kuvista nähdään kuitenkin, että luokittelu molemmissa aineistoissa perustuu pääosin samoihin tekijöihin. Ainoastaan tärkeysindeksin arvot eri muuttujien kohdalla voivat poiketa toisistaan.

Taulukossa 4.3 on kuvattuna erään puun alkua satunnaismetsässä. Vasen ja oikea oksa kertovat, millä rivillä seuraavat solmukohdat ovat kyseisen solmun muuttujaan verrattuna. Jakomuuttuja on solmukohtaan valittu aineiston muuttuja. Jakopiste puolestaan kertoo, miten solmukohdassa oleva jakomuuttujan arvot on puussa jaoteltu. Jos muuttuja on luokitteluasteikollinen, muuttujan saamat arvot on muutettu binäärisiksi. Esimerkiksi *jatkoja_luokiteltu*-muuttuja voi saada neljää eri arvoa. Taulukossa (4.3) muuttuja saa jakopisteen arvoksi 2, jolloin binäärimuuttuja on muotoa (0, 1, 0, 0), sillä $0 * 2^0 + 1 * 2^1 + 0 * 2^2 + 0 * 2^3 = 2$. Se tarkoittaa sitä, että toiseen luokkaan kuuluvat havainnot jakautuvat vasemmalle ja muihin luokkiin kuuluvat havainnot

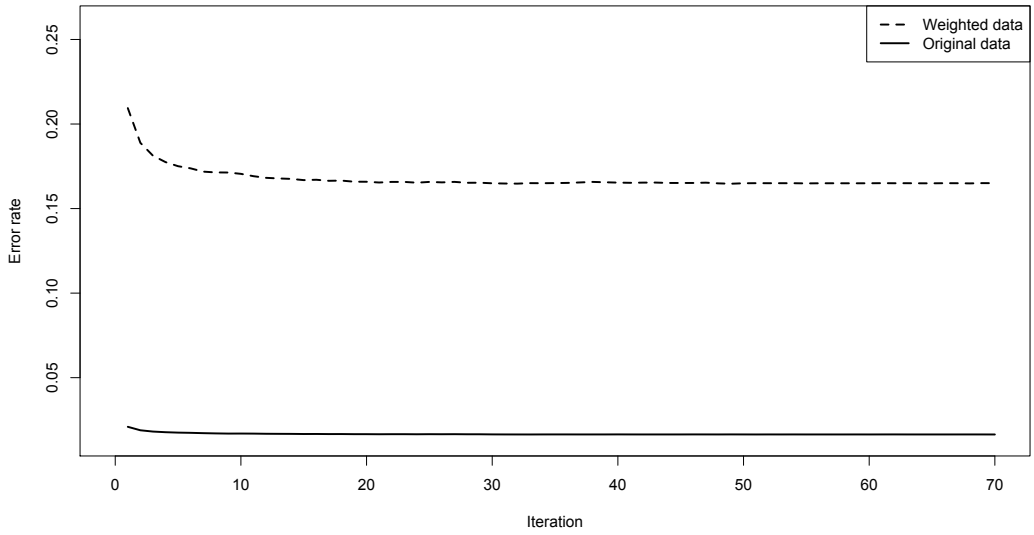


Kuvio 4.4. Painotetussa aineistossa asiakasvaihtuvuuden kannalta tärkeimmät muuttujat mallinnettuna satunnaismetsä-algoritmilla, jossa vaaka-akselilla kuvataan muuttujan tärkeysindeksin saamaa arvoa. Tarkemmat kuvaukset kuviossa mainituista muuttujista löytyvät liitteestä.

Taulukko 4.3. Esimerkki erään puun alusta satunnaismetsässä. Tarkemmat kuvaukset taulukossa mainituista muuttujista löytyvät liitteestä.

Vasen oksa	Oikea oksa	Jakomuuttuja	Jakopiste	Päättävä	Ennuste
2	3	jatkoja_luokiteltu	2	1	-
4	5	data_6kk	0.45	1	-
6	7	puhe_3kk	2.60	1	-
8	9	sopimuksen_ika_luokiteltu	7	1	-
10	11	OS	8	1	-
12	13	sopimuksen_ika_luokiteltu	12	1	-
14	15	jatkoja_luokiteltu	1	1	-
0	0	-	0	-1	1

luokitellaan oikealle. Jatkuvissa muuttujissa jakopisteen arvo jakaa jakomuuttujan vasempaan oksaan kun jakomuuttujan arvo on pienempi tai yhtä suuri kuin jakopisteen arvo. Vasemmalle jakautuvat havainnot kuuluvat todennäköisemmin poistuvien luokkaan kuin oikealle jakautuvat. Päättävä -sarake on indikaattorisarake, joka kertoo, onko kyseisellä rivillä oleva solmu päättävä. Kun sarake saa arvon -1, on kyseinen solmu päättävä solmu. Ennuste -sarakkeessa on päättävälle solmulle laskettu ennuste, joka kertoo mihin luokkaan kyseinen solmu todennäköisimmin kuuluu. Taulukon 4.3 viimeinen rivin solmu on siis päättävä solmu ja sille laskettu luokkaennuste on 1, jolloin solmu todennäköisimmin kuuluu poistuvien asiakkaiden luokkaan.



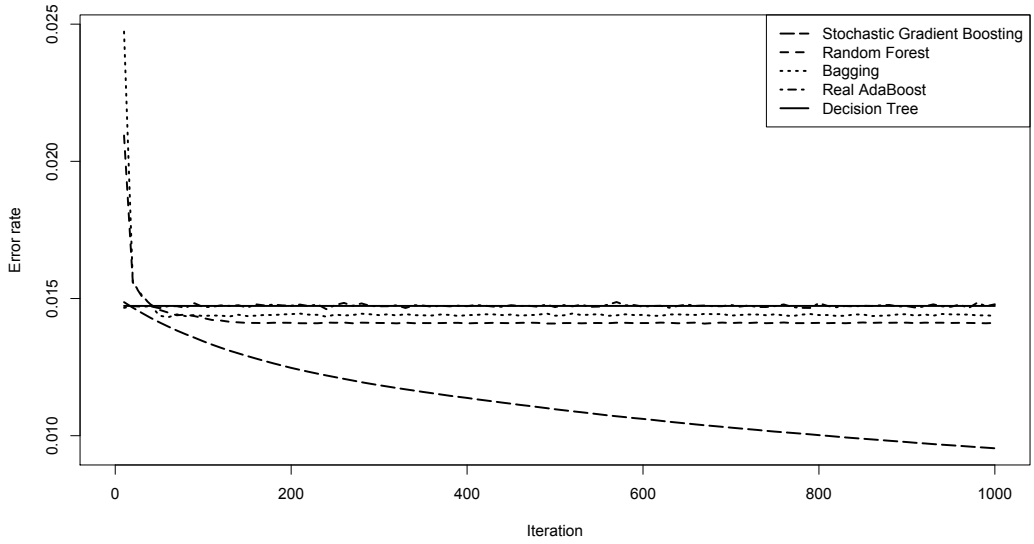
Kuvio 4.5. Painottamattomaan ja painotettuun aineistoon sovellettujen mallien luokitteluvirheet eri iterointikierroksilla. Virheet on laskettu opetusaineistosta ja mallinnukseen on käytetty satunnaismetsä-algoritmia.

Arvioitaessa satunnaismetsä-algoritmin toimintaa painotetussa ja painottamattomassa aineistossa, lasketaan jokaisella iterointikierroksella luokitteluvirhe mallille. Mallien puut on rakennettu satunnaisesti poimimalla 8 muuttujaa 54 selittävän muuttujan joukosta. Aineistoon sovitettavissa malleissa puita rakennetaan 70 kertaa ja jokaisella rakennuskerralla tehdään äänestys puiden kesken. Näin malli paranee, mitä enemmän puita tehdään, kunnes saavutetaan pienin mahdollinen luokitteluvirhe, joka yleensä saavutetaan 50 puun paikkeilla (Breiman 2001). Koska metsien rakentaminen perustuu satunnaisuuteen, on jokaisella metsällä vähän erilainen ennustevirhe. Jotta ennustevirhe olisi luotettava arvioidessa malleja keskenään, on rakennettu 50 satunnaismetsää. Kuviossa 4.5 näkyy satunnaismetsä-mallin luokitteluvirheet painottamattomassa ja painotetussa aineistoissa. Iterointikierrosten kasvaessa mallin luokitteluvirhe pienenee. Kuviosta nähdään, että painottamattomassa aineistossa luokitteluvirhe on minimissä jo 40-50 iterointikierroksen paikkeilla, kun taas painotetussa aineistossa iterointeja täytyy tehdä enemmän.

4.5 Mallinvalinta

Asiakasvaihtuvuutta analysoidessa ja ennustaessa on tärkeää, että aineistoa mallintava menetelmä luokittelee asiakkaat eli aineiston havainnot mahdollisimman hyvin. Mitä paremmin malli sovittuu aineistoon, sitä paremmin voidaan arvioida, ketkä asiakkaista ovat poistumassa seuraavana yrityksestä.

Tutkielmassa käytetyissä menetelmissä ainoastaan satunnaismetsä (*Random Fo-*

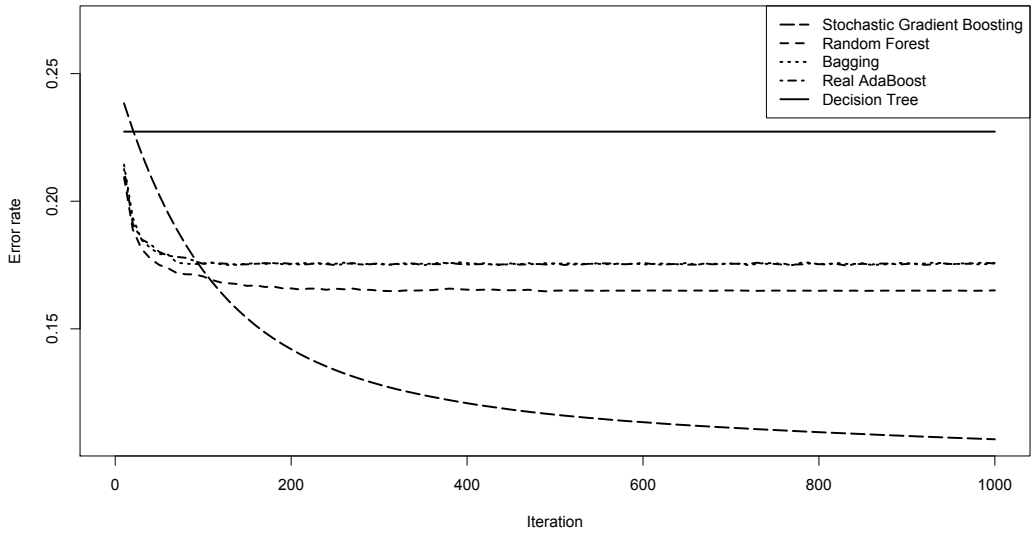


Kuvio 4.6. Käytettyjen menetelmien sekä päätöspuiden luokitteluvirheet painottamattomassa opetusaineistossa.

rest) -algoritmiin voidaan syöttää koko tutkimusaineisto. Muissa menetelmissä aineisto on jaettu satunnaisesti opetus- ja testiaineistoon siten, että 60% havainnoista kuuluu opetusaineistoon ja loput havainnoista jää testiaineistoon. Kuviossa 4.6 on esitettynä kunkin käsiteltävän mallin opetusaineiston luokitteluvirheet eri iterointikierroksilla, kun mallinnetaan painottamatonta aineistoa. Kuviosta nähdään, miten *Boosting*-menetelmiin perustuvien menetelmien (*Stochastic Gradient Boosting* ja *Real AdaBoost*) luokitteluvirheet ovat samaa tasoa päätöspuiden kanssa heti oppimisalgoritmin alussa. Tämän jälkeen erityisesti *Stochastic Gradient Boosting* -menetelmän luokitteluvirhe opetusaineistossa pienenee huomattavasti pienemmäksi kuin päätöspuun luokitteluvirhe. *Bagging* -algoritmi ja satunnaismetsä-menetelmät puolestaan saavat alkuun suhteellisen korkean luokitteluvirheen joka sitten putoaa mallin parhaalle luokittelutasolle suhteellisen nopeasti. Opetusaineistossa pienin luokitteluvirhe saavutetaan selvästi *Stochastic Gradient Boosting* -algoritmilla. Kuitenkin algoritmi tarvitsee huomattavasti enemmän toistoja kuin esimerkiksi nopea satunnaismetsä-algoritmi.

Taulukossa 4.4 esitetään menetelmien luokitteluvirheet painottamattoman aineiston testiaineistossa. Satunnaismetsä-algoritmissa luokitteluvirhe saadaan *out-of-bag* -aineiston havaintojen luokitteluvirheestä ja muissa menetelmissä luokitteluvirhe lasketaan testiaineistosta. Luokitteluvirheen perusteella selvästi paras menetelmä on *Stochastic Gradient Boosting*, vaikka satunnaismetsän luokitteluvirhe on vain 0.15 promillen päässä parhaasta menetelmästä. Voidaankin siis päätellä, että jos tarvitaan nopeasti tuloksia, niin kannattaa käyttää luokitteluvirheen mielessä satunnaismetsä-algoritmia ja muussa tapauksessa *Stochastic Gradient Boosting* -algoritmia.

Kuviossa 4.7 on piirrettynä painotettuun testiaineiston sovitettujen mallien luo-



Kuvio 4.7. Käytettyjen menetelmien sekä päätöspuiden luokitteluvirheet painotetussa opetusaineistossa.

kitteluvirheet eri iterointikierroksilla. Yhtenäinen viiva ylimpänä kuvaa päätöspuun luokitteluvirhettä. Kun aineistossa on yhtä paljon toivottuja kuin ei toivottuja havaintoja, oppimiskokonaisuuden paremmuus näkyy selvästi. Huonoimmat luokitteluvirheet saaneiden menetelmien, *Bagging* ja *Real AdaBoost*, luokitteluvirheet ovat huomattavasti alhaisempia kuin päätöspuiden luokitteluvirheet. Parhaan oppimismenetelmän, *Stochastic Gradient Boosting*, luokitteluvirhe painotetussa opetusaineistossa on kuvion 4.7 mukaan jopa puolet päätöspuiden luokitteluvirheestä.

Taulukossa 4.5 on esitetty eri menetelmien luokitteluvirheet riippumattomassa painotetussa testiaineistossa. Kuten painottamattomassa aineistossa, myös painotetussa aineistossa *Stochastic Gradient Boosting* -algoritmin avulla saavutetaan pienin luokitteluvirhe. Päätöspuiden luokitteluvirhe on tuotu myös tähän taulukkoon vertailuarvoksi. Myös painotetun aineiston luokitteluvirheistä nähdään tilastollisten oppimiskokonaisuuksien hyöty - päätöspuiden luokitteluvirhe on taulukossa mainituista menetelmistä suurin.

Taulukko 4.4. Eri oppimisyhdistelmien sekä päätöspuiden luokitteluvirheet painottamattomassa testiaineistossa.

Menetelmä	**Luokitteluvirhe ($\overline{\text{err}}$)**
Bagging	0.01440
Gradient Boosting	0.01340
Real AdaBoost	0.01365
Satunnaismetsä	0.01355
Päätöspuut, 'party'-pakkaus	0.01400

Taulukko 4.5. Eri oppimisyhdistelimen sekä päätöspuiden luokitteluvirheet painotetussa testiaineistossa.

Menetelmä	Luokitteluvirhe ($\overline{\text{err}}$)
Bagging	0.1712
Gradient Boosting	0.1685
Real AdaBoost	0.1775
Satunnaismetsä	0.1702
Päätöspuut, 'party'-pakkaus	0.2273

4.6 Aineiston painottaminen

Aineiston painottamisen ajatuksena oli saada luokittelijat luokittelemaan vahvemmin nimenomaan poistuvia asiakkaita. Painottamattomassa aineistossa vaihtuvien asiakkaiden osuus on prosentuaalisesti niin pieni, että luokittelumenetelmät tekevät luokitteluvirheitä juuri vaihtuvien asiakkaiden kohdalla. Kun opetetaan luokittelija painottamattoman aineiston testiaineistossa ja toinen luokittelija painotetussa aineistossa, voidaan testata luokittelijoiden toimivuutta painottamattoman aineiston testiaineistossa. On selvää, että painotetun aineiston luokittelija luokittelee valtavan määrän havaintoja poistuviksi, vaikka havainto olisikin pysyvä asiakas. Vastaavasti painottamattoman aineiston luokittelija toimii päinvastoin. Painotuksen mahdollisia hyötyjä voidaankin testata laittamalla painottamattoman testiaineiston luokitellut havainnot pisteytyksen perusteella paremmuusjärjestykseen. Näistä havainnoista poimitaan taulukon 4.6 mukaisesti 500 havaintoa, jotka ovat kunkin luokittelijan mukaan todennäköisimmin poistuvia havaintoja. Saaduista havainnoista voidaan laskea todellisuudessa poistuneiden osuus.

Taulukko 4.6. Todellisuudessa poistuneiden osuudet, kun eri mallien perusteella on poimittu painottamattoman testiaineiston 500 todennäköisimmin poistuvaa asiakasta.

Menetelmä	Painottamaton	Painotettu
Bagging	23,8%	38,8%
Stochastic gradient boosting	28,2%	31,0%
Real Adaboost	8,4%	23,4%
Satunnaismetsä	22,6%	45,8%

Taulukossa 4.6 on kuvattuna kunkin menetelmän luokittelijan onnistumista, kun luokittelijan avulla on poimittu 500 todennäköisimmin poistuvaa havaintoa. Taulukosta nähdään, että opetusaineiston painottamisella saavutetaan selvästi parempia onnistumisprosentteja kuin sellaisella opetusaineistolla, jota ei ole painotettu. Huomattavaa on, että *Stochastic Gradient Boosting* -algoritmilla saavutetaan painottamattomassa aineistossa paras tulos, kun arvioidaan 500 parhaan muuttujan onnistumisen todennäköisyyttä. Kuitenkaan menetelmällä ei saavuteta parhaita tuloksia painotetulla aineistolla, vaan satunnaismetsä-algoritmilla saavutetaan huomattavasti parempia tuloksia. Erityisesti mielenkiintoiseksi tämän tuloksen tekee se, että

painotetun aineiston luokitteluvirhettä tarkastellessa *Stochastic Gradient Boosting* -algoritmi osoittautui parhaaksi menetelmäksi kyseisessä aineistossa.

5 Johtopäätökset

Tutkielman tavoitteena oli selvittää parhaat tilastolliset menetelmät, joilla voitaisiin ennustaa asiakasvaihtuvuutta. Aineistona käytettiin suomalaisen operaattorin tarjoamia anonymisoituja asiakastietoaineistoja, joista toinen oli painottamaton ja toinen painotettu. Painottamaton aineisto sisälsi hyvin vähän poistuvia havaintoja, kun taas painotetussa aineistossa poistuvia asiakkaita ja pysyviä asiakkaita oli yhtä paljon. Asiakasvaihtuvuuden mallintamiseen käytettiin tilastollisia oppimisyhdistelmiä, jotka pohjautuvat luokittelu- ja regressiopuihin *(CART)*. Tilastollisten oppimisyhdistelmien etuna verrattuna esimerkiksi päätöspuuhun on se, että menetelmillä voidaan saavuttaa päätöspuita pienemmät luokitteluvirheet ja sitä kautta paremmat asiakasvaihtuvuusennusteet.

Tutkielmassa tarkasteltiin neljää eri oppimiskokonaisuusmenetelmää ja niiden sopivuutta asiakasvaihtuvuuden ennustamiseen. Menetelmiä sovellettiin niin painottamattomaan kuin painotettuunkin aineistoon. Menetelmien absoluuttista luokitteluvirheitä sekä parhaan kymmenyksen nosteen tuottamien havaintojen luokitteluvirhettä vertailemalla saatiin laitettua menetelmät paremmuusjärjestykseen ja sitä kautta pystyttiin valitsemaan tutkielman aineistoihin parhaiten soveltuvat menetelmät. Menetelmien tehokkuuden mittarina käytettiin päätöspuita. Huomattavaa on, että painottamattoman aineiston vähäinen suotuisten tapausten lukumäärä aiheuttaa pienen luokitteluvirheen asettamalla kaikki havainnot pysyviksi asiakkaiksi. Tämän vuoksi painottamattomaan aineistoon sovellettaessa *Bagging*-menetelmää, tulokset eivät olleen yhtään parempia kuin päätöspuilla saadut tulokset. Painotetussa aineistossa puolestaan kaikki tilastolliset oppimiskokonaisuusmenetelmät saivat huomattavasti pienempiä luokitteluvirheitä kuin päätöspuut.

Muuttujat ovat myös mielenkiintoisia asiakasvaihtuvuuden kannalta. On mielekästä tietää, millaisia asiakkaita poistuvat yleisesti ovat. Kaikki kolme eri menetelmää, joissa on mahdollisuus antaa muuttujille tärkeyspisteet, valikoivat pääsääntöisesti samoja muuttujia. Menetelmistä muuttujanvalintaa voidaan tehdä *Real AdaBoost*, *Stochastic Gradient Boosting* ja satunnaismetsä -algoritmeilla.

Kun menetelmiä arvioitiin luokitteluvirheellä, parhaimmaksi menetelmäksi osoittautui *Stochastic Gradient Boosting* -algoritmi. Menetelmä oli selkeästi paras algoritmi niin painottamattomassa kuin painotetussa aineistossa. Menetelmän heikkoutena on puolestaan sen hitaus. *Stochastic Gradient Boosting* -algoritmi tarvitsee moninkertaisen määrän iterointikierroksia verrattuna vaikkapa toiseksi parhaaseen ja nopeaan menetelmään, satunnaismetsä-algoritmiin.

Aineiston painottamisella saavutettiin mielenkiintoisia tuloksia. Testattaessa painotetun aineiston luokittelijaa painottamattoman aineiston testiaineistoon luokittelijat luokittelevat havainnot herkästi poistuviksi. Luokitellessa havainnolle annetaan myös poistumatodennäköisyys ja kun todennäköisyyksien perusteella järjestetään havainnot järjestykseen, voidaan niihin soveltaa parhaan kymmenyksen nostetta. Painottamattoman testiaineiston poistuvien havaintojen vähäisyyden vuoksi parhaan kymmenyksen nosteessa poimittiin ainoastaan 500 luokittelijan mukaan todennäköi-

simmin poistuvaa havaintoa. Kun näiden havaintojen todellista poistuvien osuutta tarkasteltiin, olivat painotetun aineiston luokittelijat selkeästi parempia kuin painottamattoman aineiston luokittelijat. Lisäksi huomattiin, että luokitteluvirheen mielessä paras algoritmi, *Stochastic Gradient Boosting*, ei ollutkaan parhaan kymmenyksen nosteen avulla tarkastellessa parhaita tuloksia antava menetelmä. Sen sijaan parhaat tulokset, eli yli 45% tarkkuus, saavutettiin satunnaismetsä-algoritmin luokittelijalla, joka opetettiin painotetulla aineistolla.

Mallien ennustamisen luotettavuuden kannalta olisi mielenkiintoista laajentaa aineistoja osa-aineistoista operaattorin koko liittymäkannan sisältävään aineistoon. Keskimäärin suomalaisella operaattorilla on noin 3 190 000 asiakasta (kuvio 1.1). Tämä tarkoittaisi sitä, että aineistossa olisi karkeasti 40 kertaa enemmän havaintoja, kuin tutkielmassa käytettävissä olevissa aineistoissa. Se vaatisi, että tietokoneen laskentatehon pitäisi olla erittäin hyvällä tasolla, jotta laskentaintensiiviset tilastolliset oppimisalgoritmit saataisiin toimimaan. Lisäksi ennustavuuden tarkkuutta olisi mielenkiintoista selvittää myös pitemmällä aikavälillä, jolloin voitaisiin saada selville, miten asiakkaiden käyttäytyminen eri vuodenaikoina vaikuttaa mallin ennustamisen luotettavuuteen.

Lähteet

Breiman, L., Friedman, J. H., Olshen, R. A. & Stone, C. J. (1984), "Classification and Regression Trees", Wadsworth Belmont CA.

Breiman, L. (1996), "Bagging Predictors", *Machine Learning*, 24, 123-140, Kluwer Academic Publishers, Boston.

Breiman, L. (2001), "Random Forests", *Machine Learning*, 45, 5-32, Kluwer Academic Publishers, Boston.

Breiman, L. Cutler, A., Liaw, A. & Wiener, M. (2010), "Package 'randomForest'". [Viitattu 10.9.2013]. Saatavana: http://cran.r-project.org/web/packages/randomForest/index.html.

Clup, Mark., Johnson, K. & Michailidis, G. (2012), "Package 'ada'". [Viitattu 10.9.2013]. Saatavana: http://cran.r-project.org/web/packages/ada/index.html.

Dna Oy (2006–2012), "Osavuosikatsaukset"[Viitattu 1.12.2013]. Saatavana: http://www.dna.fi/dna-oy/tilinpaatokset-ja-osavuosikatsaukset

Efron, B. & Tibshirani, R. (1993), "An Introduction to the Bootstrap", Chapman & Hall, New York.

Elisa Oyj (2006–2012), "Osavuosikatsaukset"[Viitattu 1.12.2013]. Saatavana: http://corporate.elisa.fi/elisa-oyj/sijoittajille/taloustiedot/tilinpaatos-osavuosikatsaukset-ja-materiaalit/

Franses, P. H. & Paap, R. (2004) "Quantitative Models in Marketing Research", Cambridge University Press, UK.

Friedman, J. H. (1999), "Greedy Function Approximation: A Gradient Boosting Machine", *Annals of Statistics*, 29, 1189-1232, Institute of Mathematical Statistics, USA.

Friedman, J. H., Hastie, T. & Tibshirani, R. (2000), "Additive logistic regression: a statistical view of boosting" *The Annals of Statistics*, (Vol. 28, No. 2), 337-407, Institute of Mathematical Statistics, USA.

Friedman, J. H. (2002), "Stochastic Gradient Boosting", *Computational Statistics and Data Analysis*, 38, 367-378, Elsevier, USA.

Genuer, R., Poggi, J.-M. & Tuleau-Malot, C. (2010), "Variable Selection Using Random Forest" *Pattern Recognition Letters*, (Vol. 31, no. 14), 2225-2236, Elsevier, North-Holland.

Hastie, T., Tibshirani, R. & Friedman, J. (2011), "The Elements of Statistical Learning: Data Mining, Inference and Prediction", Springler-Verlag, New York.

Hastie, T., & Tibshirani, R. (1990), "Generalized additive models", Chapman & Hall, London.

Kuluttajabarometri (2013), "Eri laitteiden ja yhteyksien yleisyys kotitalouksissa, toukokuu 2013", Helsinki: Tilastokeskus. [Viitattu 16.7.2013]. Saatavana: http://www.stat.fi/til/kbar/2013/06/kbar_2013_06_2013-06-27_kuv_012_fi.html.

Kuncheva, L. & Whitaker, C. (2003), "Measures of diversity in classifier ensembles", *Machine Learning*, 51, 181-207, Kluwer Academic Publishers, Boston.

Lemmens, A. & Croux, C. (2006), "Bagging and boosting classification trees to predict churn", *Journal of Marketing Research*, (Vol. 43, No. 2), 276-286, American Marketing Association, Chicago.

Lunetta, K. L., Hayward, L. B., Segal, J. & Eerdewegh, P. V. (2004), "Screening large-scale association study data: exploiting interactionsusing random forests", *BMC Genetics*, 5:34, BioMed Central Ltd.

Nelsin, S. A., Gupta, S., Kamakura, W., Lu, W. & Mason, C. H. (2006), "Defection Detection: Measuring and Understanding the Predictive Accuracy of Customer Churn Models", *Journal of Marketing Research*, 43, 204-211, American Marketing Association, Chicago.

Opitz, D. & Maclin, R. (1999) "Popular Ensemble Menthods: An Empirical Study", *Journal of Artificial Intelligence Research*, 11, 169-198, AI Access Foundation, USA.

Paukkeri, M.-S., Väyrynen, J., Arppe, A. (2012), "Exploring Extensive Linguistic Feature Sets in Near-synonym Lexical Choice", *Lecture Notes in Computer Science*, 7182, 1-12, Springer-Verlag.

Peters, A., Hothorn, T., Ripley, B. D., Therneau, T. & Atkinson, B. (2013), "Package 'ipred'". [Viitattu 10.9.2013]. Saatavana: http://cran.r-project.org/web/packages/ipred/index.html.

R Core Team (2013), "R: A Language and Environment for Statistical Computing", R Foundation for Statistical Computing, Vienna, Austria. [Viitattu 10.9.2013]. http://www.R-project.org.

Ridgeway, G. (2013), "Package 'gbm'". [Viitattu 10.9.2013]. Saatavana: http://cran.r-project.org/web/packages/gbm/index.html.

Schapire, R. E., Freund, Y., Bartlett, P. & Lee, W. S. (1998), "Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods", *Annals of Statistics*, 26, 1651-1686, Institute of Mathematical Statistics, USA.

Schapire, R. E. & Signer, Y. (1999), "Improved boosting algorithms using confidence-rated predictions", *Machine Learning*, 37, 297-336, Kluwer Academic Publishers, Boston.

TeliaSonera Finland Oyj (2006–2012), "Osavuosikatsaukset"[Viitattu 1.12.2013]. Saatavana: http://www.teliasonera.com/fi/sijoittajat/raportit-ja-presentaatiot/6398/

Väestörakenne (2012), "Vuoden 2012 väkiluvun kasvusta vieraskielisten osuus 87 prosenttia", Helsinki: Tilastokeskus. [Viitattu 16.7.2013]. Saatavana http://tilastokeskus.fi/til/vaerak/2012/vaerak_2012_2013-03-22_tie_001_fi.html?ad=notify

Liite: Aineistojen muuttujat

Taulukko 1. Tutkielmassa mainitut muuttujat selityksineen.

Muuttuja	Selite
sex	Sukupuoli
ika	Liittymän omistajan ikä
ika_1	Liittymän omistajan ikä luokiteltuna
ika_2	Liittymän omistajan ikä sukupuolen ja iän mukaan luokiteltuna
PRODUCT_GRP_NAME	Liittymän ryhmitelty tuoteluokka
POSTAL_CODE_AREA	Ryhmitelty asiakkaan postialue
OS	Puhelimen käyttöjärjestelmä
jatkoja	Perättäisten sopimusten lukumäärä
jatkoja_luokiteltu	Luokiteltu perättäisten sopimusten lukumäärä
lasku	Liittymän keskimääräinen lasku
lasku_luok	Luokiteltu liittymän keskimääräinen lasku
paate_ika	Käytettävän päätelaitteen ikä
sopimus_päättyy	Indikaattorimuuttuja: päättyykö sopimus tarkastellulla ajanjaksolla
sopimuksen_ika	Sopimuksen ikä
sopimuksen_ika_luokiteltu	Luokiteltu sopimuksen ikä
n_3kk	Puhelujen keskimääräinen lukumäärä viimeisen 3 kk aikana
n_6kk	Puhelujen keskimääräinen lukumäärä viimeisen 6 kk aikana
n_12kk	Puhelujen keskimääräinen lukumäärä viimeisen 12 kk aikana
n_1to3	Puhelujen lukumäärän muutos: *n_3kk* jaettuna viime kuun puhelujen lukumäärällä
n3_luokiteltu	Luokiteltu *n_3kk*
n_1to3_luok	Luokiteltu *n_1to3*
puhe_3kk	Keskimääräiset soitetut minuutit viimeisen 3 kk aikana
puhe_6kk	Keskimääräiset soitetut minuutit viimeisen 6 kk aikana
puhe_12kk	Keskimääräiset soitetut minuutit viimeisen 12 kk aikana
puhe_1to3	Soitettujen minuuttien määrän muutos: *puhe_3kk* jaettuna viime kuun soitettujen minuuttien määrällä
puhe3_luokiteltu	Luokiteltu *puhe_3kk*
data_3kk	Keskimääräinen datan käytön määrä viimeisen 3 kk aikana
data_6kk	Keskimääräinen datan käytön määrä viimeisen 6 kk aikana
data_12kk	Keskimääräinen datan käytön määrä viimeisen 12 kk aikana
data_1to3	Datan käytön määrän muutos: *data_3kk* jaettuna viime kuun datan käytön määrällä
data_1to3_luok	Luokiteltu *data_1to3*
muutos_n	Puhelujen lukumäärien muutos: 4-2 viime kuukauden keskiarvo miinustettuna viime kuun puhelujen lukumäärällä
muutos_puhe	Puheminuuttien muutos: 4-2 viime kuukauden keskiarvo miinustettuna viime kuun puheminuuteilla